Communication Protocols and Applications for ZigBee-Based Wireless Sensor Networks

Meng-Shiuan Pan and Yu-Chee Tseng
Department of Computer Science
National Chiao Tung University
Hsin-Chu, Taiwan
Email:{mspan, yctseng}@cs.nctu.edu.tw

***Abstract*—ZigBee is a standard which is considered to be suitable for *wireless sensor networks (WSNs)*. This paper discusses four research topics in ZigBee-based WSNs. First, we observe that using the ZigBee network formation scheme may lead to some nodes cannot join the network. We model an *orphan problem* to characterize this phenomenon. Second, based on the observation that convergecast is a fundamental operation in WSNs. We define a *minimum delay beacon scheduling problem* for quick convergecast in ZigBee tree-based WSNs. Third, we promote a new concept of *long-thin (LT) topology* for WSNs, where a network may have a number of linear paths of nodes as backbones connecting to each other. We propose a simple, yet efficient, address assignment scheme for a LT WSN. Finally, we introduce a novel 3D dynamically path finding algorithm based on ZigBee WSNs. This algorithm aims to guide people to safe places when emergencies happen.**

***Index Terms*—graph theory, IEEE 802.15.4, pervasive computing, wireless sensor network, ZigBee**

I. Introduction

The rapid progress of wireless communication and embedded micro-sensing MEMS technologies has made wireless sensor networks possible. A WSN consists of many inexpensive wireless sensors capable of collecting, storing, processing environmental information, and communicating with neighboring nodes. Applications of WSNs include wildlife monitoring [2][3], object tracking [11], and so on.

Recently, several WSN platforms have been developed, such as MICA [4] and Dust Network [1]. For interoperability among different systems, standards such as ZigBee [17] have been developed. In the ZigBee protocol stack, physical and MAC layer protocols are adopted from the IEEE 802.15.4 standard [9]. ZigBee solves interoperability issues from the physical layer to the application layer.

ZigBee supports three kinds of networks, namely *star*, *tree*, and *mesh* networks. A *ZigBee coordinator* is responsible for initializing, maintaining, and controlling the network. A star network has a coordinator with devices directly connecting 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 the associating with the coordinator or a router. In a tree network, the coordinator and routers can announce beacons. However, in a mesh network, regular beacons are not allowed. Beacons are an important mechanism to support power management. Therefore, the tree topology is preferred, especially when energy saving is a desirable feature.

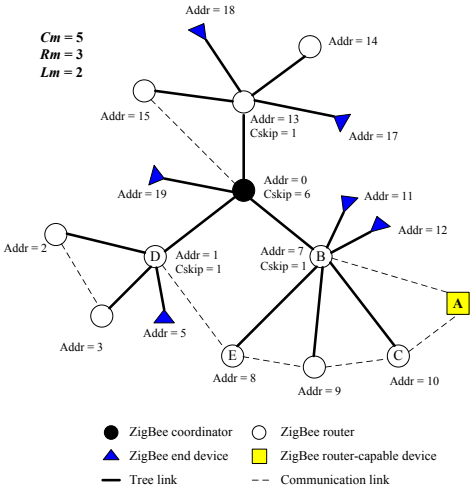


Fig. 1. An example ZigBee tree network.

In ZigBee, a device is said to join a network successfully if it can obtain a network address from the coordinator or a router. Before forming a network, the coordinator determines 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 $Cm \geq Rm$. ZigBee specifies a distributed address assignment using parameters Cm, Rm, and Lm to calculate nodes' network addresses. While these parameters facilitate address assignment, they also prohibit a node from joining a network. We say that a node becomes an *orphan node* when it can not associate with the network but there are unused address spaces. 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.

Considering that data gathering is a major application of

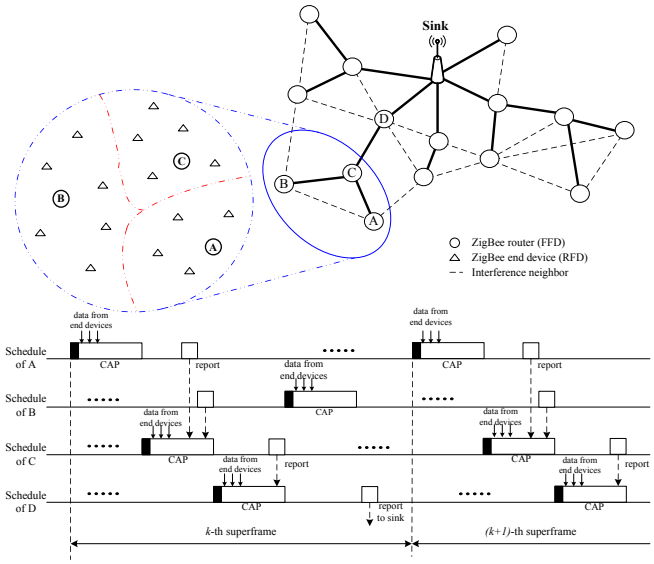


Fig. 2. An example of convergecast in a ZigBee tree-based network.

WSNs, *convergecast* has been investigated in several works [5][7]. This paper discusses efficient convergecast solutions for WSNs that are compliant with the ZigBee/IEEE 802.15.4 standards. Assuming a tree topology, Fig. 2 shows the problem scenario. The network contains one *sink* (ZigBee coordinator), some *full function devices* (ZigBee routers), and some *reduced function devices* (ZigBee end devices). Each ZigBee router is responsible for collecting sensed data from end devices associated with it and relaying incoming data to the sink. According to the ZigBee specification, a ZigBee router can announce a beacon to start a superframe. Each superframe consists of an *active portion* followed by an *inactive portion*. On receiving its parent router's beacon, an end device has also to wake up for an active portion to sense the environment and communicate with its parent device. However, to avoid collision with its neighbors, a router should shift its active portion by a certain amount. Fig. 2 shows a possible allocation of active portions for routers A, B, C, and D. The collected sensory data of A in the k-th superframe can be sent to C in the k-th superframe. However, because the active portion of B in the k-th superframe appears after that of C, the collected data of B in the k-th superframe can only be relayed to C in the $(k+1)$-th superframe. The delay can be eliminated if the active portion of B in the k-th superframe appears before that of C. The delay is not negligible because of the low duty cycle design of IEEE 802.15.4. For example, in 2.4 GHz PHY, with 1.56% duty cycle, a superframe can be up to 251.658 seconds (with an active portion of 3.93 seconds). Clearly, for large-scale WSNs, the convergecast latency could be significant if the problem is not carefully addressed. The quick convergecast problem is to schedule the beacons of routers to minimize the convergecast latency.

Next, we discuss the *long-thin (LT)* network topology, which seems to have a very specific architecture, but may be commonly seen in many WSN deployments in many applications, such as gas disclosure detection of fuel pipes, carbon dioxide concentration monitoring in tunnels, stage measurements in sewers, and so on. In such a network, nodes may form several long backbones and these backbones are to extend the network to the intended coverage areas. A backbone is a linear path which may contain tens or hundreds of ZigBee routers and may go beyond hundreds or thousands of meters. So the network area can be scaled up easily with limited hardware cost. While the ZigBee address assignment scheme has low complexity, it also prohibits the network from scaling up and thus can not be used in LT networks. An efficient address assignment scheme for ZigBee-based LT WSNs will be presented.

In this paper, we introduce a novel 3D dynamically path finding algorithm based on ZigBee WSNs. This service aims to guide people to safe places when emergencies happen. At normal time, the network is responsible for monitoring the environment. When emergency events are detected, the network can adaptively modify its topology to ensure transportation reliability, quickly identify hazardous regions that should be avoided, and find safe navigation paths that can lead people to exits. In particular, the structures of 3D buildings are taken into account in our design. Prototyping results will be demonstrated, which show that our protocols can react to emergencies quickly at low message cost and can find safe paths to exits.

The rest of this paper is organized as follows. Preliminaries are given in Section II. Section III and Section IV introduce the orphan and convergecast problems in ZigBee-based WSNs, respectively. Section V presents the proposed address assignment scheme for LT WSNs. Section VI presents the proposed emergency guiding service based on ZigBee WSNs. Finally, Section VII concludes this paper.

II. Preliminaries

A. ZigBee Address Assignment

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 $2^{16} = 65536$, so the coordinator can not decide the Cm, Rm, and Lm arbitrarily. Moreover, the above assignment is much suitable for regular networks, but not for LT WSNs. For example, when setting $Cm = 4$ and $Rm = 2$, the depth of the network can only be 14. Also, when there are some LT backbones, the address space will not be well utilized.

B. IEEE 802.15.4 Superframe Structure

The ZigBee coordinator defines the superframe structure of a ZigBee network. The structure of superframes is controlled by two parameters: *beacon order* (BO) and *superframe order* (SO), which decide the lengths of a superframe and its active potion, respectively. For a beacon-enabled network, the setting of BO and SO should satisfy the relationship $0 \le SO \le BO \le 14$. (A non-beacon-enabled network should set $BO = SO = 15$ to indicate that superframes do not exist.) Each active portion consists of 16 equal-length slots, which can be further partitioned into a *contention access period* (CAP) and a *contention free period* (CFP). The CAP may contain the first i slots, and the CFP contains the rest of the $16 - i$ slots, where $1 \le i \le 16$. Slotted CSMA/CA is used in CAP. FFDs which require fixed transmission rates can ask for *guarantee time slots (GTSs)* from the coordinator. A CFP can support multiple GTSs, and each GTS may contain multiple slots. Note that only the coordinator can allocate GTSs. After the active portion, devices can go to sleep to save energy.

In a beacon-enabled star network, a device only needs to be active for $2^{-(BO-SO)}$ portion of the time. Changing the value of $(BO - SO)$ allows us to adjust the on-duty time of devices. However, for a beacon-enabled tree network, routers have to choose different times to start their active portions to avoid collision. Once the value of $(BO-SO)$ is decided, each router can choose from 2^{BO-SO} slots as its active portion. In the revised version of IEEE 802.15.4 [10], a router can select one active portion as its *outgoing superframe*, and based on the active portion selected by its parent, the active portion is called its *incoming superframe*. In an outgoing/incoming superframe, a router is expected to transmit/receive a beacon to/from its child routers/parent router. When choosing a slot, neighboring routers' active portions (i.e., outgoing superframes) should be shifted away from each other to avoid interference. But, the specification does not clearly define how to choose the locations of routers' active portions such that the converge-cast latency can be reduced. In our work, we consider two kinds of interference between routers. Two routers have *direct interference* if they can hear each others' beacons. Two routers have *indirect interference* if they have at least one common neighbor. Both interferences should be avoided when choosing routers' active portions.

III. The Orphan Problem in ZigBee-Based WSNs

Given a sensor network, we model the orphan problem in ZigBee by two subproblems. In the first 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. We are also given parameters Cm, Rm, and Lm such that $Cm \ge Rm$. The goal is to assign parent-child relationships to nodes such that as many vertices in V_r can join the network as possible. Below, we translate the subproblem to a tree formation problem.

Definition 1: Given $G_r = (V_r, E_r)$, Rm, Lm, and an integer $N \le |V_r|$, 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 [8], it is shown that the *Degree-Constrained Spanning Tree (DCST)* as defined below is NP-complete.

Definition 2: Given $G = (V, E)$ and a positive integer $K \le |V|$, the *Degree-Constrained Spanning Tree (DCST)* problem is to find a spanning tree T from G such that no vertex in T has a degree larger than K.

Theorem 1: The BDDTF problem is NP-complete.

Proof: We can reduce the DCST problem to a special case of the BDDTF problem in polynomial time. The details can be found in [14]. □

In the second subproblem, we will connect non-router-capable devices to the tree T constructed earlier such that the ZigBee definition is followed and as many end devices are connected to T as possible. Toward this goal, we model the sensor network by a graph $G_d = (\{\hat{V_r} \cup V_e\}, E_d)$, where $\hat{V_r}$ consists of all routers in T formed in the first stage, 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 $C_v \ge (Cm - Rm)$ end devices. From G_d, we construct a 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 $\tilde{E_d}$.

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. Intuitively, we duplicate each $v \in \hat{V_r}$ by C_v vertices. Then we enforce each vertex in $\tilde{V_r}$ to be connected to at most one vertex in $\tilde{V_e}$. This translates the problem to a maximum matching problem in graph $\tilde{G_d}$.

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 $\tilde{G_d}$.

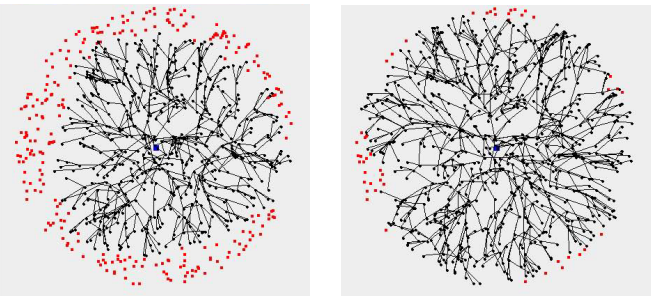
In the above, we prove that the first subproblem is intractable. On the contrary, the maximum matching problem in Definition 3 is solvable in polynomial time. Based on the above model, we design a two-stage network formation policy to relieve the orphan problem. The first stage algorithm is designed for the BDDTF problem and the goal is to connect as many routers as possible. And then, based on the result of the first stage, the second stage algorithm, which is designed for the EDMM problem, is used to reduce the number of orphan end devices.

Below, we present a centralized algorithm for the BDDTF problem. Given a graph $G_r = (V_r, E_r)$, the algorithm finds a tree $T = (V_T, E_T)$ from G_r which satisfies the ZigBee tree definition. In this algorithm, we decide to connect or disconnect a node according to its *association priority*. The priority assignment is based on forming BFS trees from G_r: (i) A node x has a higher priority than another node y if the subtree rooted at x (in the BFS tree) has more nodes than the subtree rooted at y. (ii) If the subtrees rooted at nodes x and y have the same number of nodes, the one with a less number of potential parents has a higher priority. A node takes a tree neighbor as its potential parent if this neighbor has a smaller hop count distance to the root of the BFS tree than its. This algorithm consists of a sequence of iterations.

1) Initially, T contains only the coordinator t.
2) In each iteration, there are two phases.
   - *Span phase*: pick a node in T, say x, and span from x a subtree T' to include as many nodes not in the tree T as possible. Then we attach T' to T to form a larger tree T.
   - *Prune phase*: traverse nodes in T' from x in a top-down manner to trim T. When visiting a node, say y, if y has more than Rm children, we will compute their priorities based on T'. Only the Rm highest prioritized children will remain in T', and the other children will be pruned from T'.
3) The resulting tree is then passed to the next iteration for another span and prune phases.

After finishing the first stage algorithm, we will have a $\tilde{G_d} = (\{\tilde{V_r} \cup \tilde{V_e}, \tilde{E_d}\})$. The algorithm for the EDMM problem is to apply a bipartite maximum matching algorithm in [6].

Below, we present a large-scale simulation result in a circular field of a radius 200 m and a coordinator at the center. There are 800 router-capable devices randomly deployed in the network. The transmission range of nodes is set to 35 m. We set $Cm = Rm = 3$ and $Lm = 7$, which implies that this network can accommodate up to 3280 routers. We compare the proposed first stage algorithm with the ZigBee network formation algorithm. The simulation result shows that, when using ZigBee, more than 25% of devices (about 207.45 devices) will become orphan nodes. When use the proposed algorithm, in average, about 65.8 devices can not join the network. Fig. 3 shows one simulation scenario. We can observe that when using ZigBee, many devices near the network boundary can not join the network and some devices near the center do not have any child. For more simulation results, readers can refer to [14].



(a) (b)

Fig. 3. Network formation results by (a) ZigBee and (b) the proposed first stage algorithms.

IV. Quick Convergecast in ZigBee Beacon-Enabled Tree WSNs

This section introduces the quick convergecast problem in ZigBee beacon-enabled tree networks. Given a ZigBee network, we model it by a graph $G = (V, E)$, where V contains all routers and the coordinator and E contains all symmetric communication links between nodes in V. The coordinator also serves as the sink of the network. End devices can only associate with routers, but are not included in V. From G, we can construct an *interference graph* $G_I = (V, E_I)$, where edge $(i, j) \in E_I$ if there are direct/indirect interferences between i and j. There is a duty cycle requirement α for this network. From α, we can determine the most appropriate value of $BO - SO$. We denote by $k = 2^{BO-SO}$ the number of active portions (or slots) per beacon interval.

The beacon scheduling problem is to find a slot assignment $s(i)$ for each router $i \in V$, where $s(i)$ is an integer and $s(i) \in [0, k-1]$, such that router i's active portion is in slot $s(i)$ and $s(i) \neq s(j)$ if $(i, j) \in E_I$. Here the slot assignment means the position of the outgoing superframe of each router (the position of the incoming superframe, as clarified earlier, is determined by the parent of the router). Motivated by Brook's theorem [16], which proves that n colors are sufficient to color any graph with a maximum degree of n, we would assume that $k \geq D_I$, where D_I is the maximum degree of G_I.

Given a slot assignment for G, the report latency from node i to node j, where $(i, j) \in E$, is the number of slots, denoted by d_{ij}, that node i has to wait to relay its collected sensory data to node j, i.e.,

$$d_{ij} = (s(j) - s(i)) \bmod k. \tag{2}$$

Note that the report latency from node i to node j (d_{ij}) may not by equal to the report latency from node j to node i (d_{ji}). Therefore, we can convert G into a weighted directed graph $G_D = (V, E_D)$ such that each $(i, j) \in E$ is translated into

two directed edges (i, j) and (j, i) such that $w((i, j)) = d_{ij}$ and $w((j, i)) = d_{ji}$. The report latency for each $i \in V$ to the sink is the sum of report latencies of the links on the shortest path from i to the sink in G_D. The latency of the convergecast, denoted as $L(G)$, is the maximum of all nodes' report latencies.

Definition 4: Given $G = (V, E)$, G's interference graph $G_I = (V, E_I)$, and k available slots, the *Minimum Delay Beacon Scheduling (MDBS) problem* is to find an interference-free slot assignment $s(i)$ for each $i \in V$ such that the convergecast latency $L(G)$ is minimized.

To prove that the MDBS problem is NP-complete, we define a decision problem as follows.

Definition 5: Given $G = (V, E)$, G's interference graph $G_I = (V, E_I)$, k available slots, and a delay constraint d, the *Bounded Delay Beacon Scheduling (BDBS) problem* is to decide if there exists an interference-free slot assignment $s(i)$ for each $i \in V$ such that the convergecast latency $L(G) \leq d$.

Theorem 2: The BDBS problem is NP-complete.

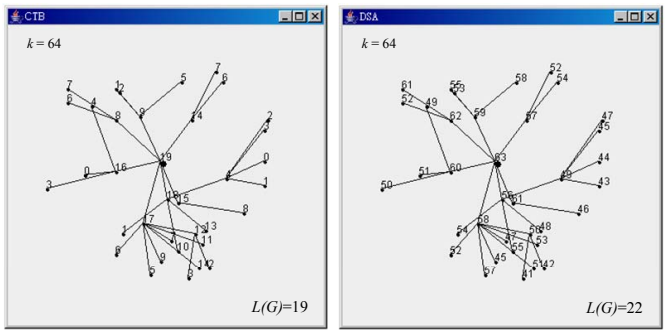
Proof: We can reduce the 3-CNF-SAT problem to a special case of the BDBS problem in polynomial time. The details can be found in [15] □

In this paper, we propose a centralized slot assignment algorithm for the MDBS problem. Given $G = (V, E)$, $G_I = (V, E_I)$, and k, our algorithm is composed of the following three phases:

1) From G, we first construct a BFS tree T rooted at sink t.
2) We traverse vertices of T in a bottom-up manner. For each vertex v visited, we first compute a temporary slot number $t(v)$ for v as follows.
   a) If v is a leaf node, we set $t(v)$ to the minimal non-negative integer l such that for each vertex u that has been visited and $(u, v) \in E_I$, $(t(u) \bmod k) \neq l$.
   b) If v is an in-tree node, let m be the maximum of the numbers that have been assigned to v's children, i.e., $m = \max\{t(child(v))\}$, where $child(v)$ is the set of v's children. We then set $t(v)$ to the minimal non-negative integer $l > m$ such that for each vertex u that has been visited and $(u, v) \in E_I$, $(t(u) \bmod k) \neq (l \bmod k)$.

   After every vertex v is visited, we make the assignment $s(v) = t(v) \bmod k$.
3) In this phase, vertices are traversed sequentially from t in a top-down manner. When each vertex v is visited, we try to greedily find a new slot l such that $(s(par(v)) - l) \bmod k < (s(par(v)) - s(v)) \bmod k$, such that $l \neq s(u)$ for each $(u, v) \in E_I$, if possible. Then we reassign $s(v) = l$.

Note that in phase 2, the number $t(v)$ is not a modulus number. However, we will check that if $t(v)$ is converted to a slot number, no interference will occur. Intuitively, this is a temporary slot assignment that will incur the least latency to v's children. At the end, $t(v)$ is converted to a slot assignment $s(v)$. Phase 3 is a greedy approach to further reduce $L(G)$. Moreover, we also design a distributed slot assignment algorithm in [15].



(a) (b)

Fig. 4. Slot assignment results by the proposed (a) centralized and (b) distributed slot assignment schemes.

Fig. 4 shows some slot assignment results of centralized and distributed slot assignment schemes when $r = 35\ m$ and $k = 64$. Devices are randomly distributed. The transmission range of routers is set to $20\ m$. We can observe that the centralized method performs better than the distributed one. More simulation results can also be found in [15].

V. An Address Assignment Scheme for ZigBee-Based LT Networks

In this work, we propose an address assignment scheme for ZigBee-based LT WSNs. Our goal is to automatically form a LT WSN and give addresses to nodes. Assuming that all nodes are router-capable devices, Fig. 5 shows an example of a LT WSN. Nodes are divided into multiple *clusters*, each as a line segment. For each cluster, we define two special nodes, named *cluster head* and *bridge*. The cluster head (resp., the bridge) is the node that has the smallest (resp., largest) hop count to the coordinator. As a special case, the coordinator, is also considered as a cluster head. The other nodes are *network nodes* (refer to Fig. 5). A cluster C is a *child* cluster of a cluster C' if the cluster head of C is connected to the bridge of C'. Reversely, C' is C's *parent* cluster. Note that a cluster must have a linear path as its subgraph. But it may have other extra links beside the linear path. For example, in Fig. 5, there are two extra radio links (A, $A2$) and ($A1$, $A3$) in A's cluster. To be compliant with ZigBee, we divide the ZigBee 16-bit network address into two parts, an m-bit *cluster ID* and a $(16 - m)$-bit *node ID*. The network address of a node v is thus expressed as (C_v, N_v), where C_v and N_v are v's cluster ID and node ID, respectively.

Before deploying a network, the network manager should carefully plan the placement of cluster head, bridge, and network nodes. There are some basic principles:

1) The network contains a number of linear path.
2) For each cluster, the first and the last nodes are pre-assigned (manually) as cluster head and bridge, respectively.

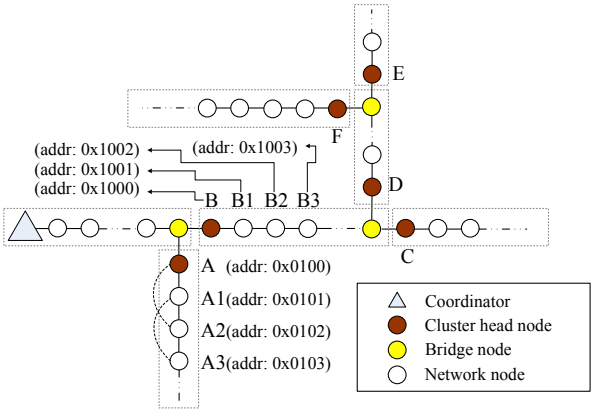


Fig. 5. A LT WSN.

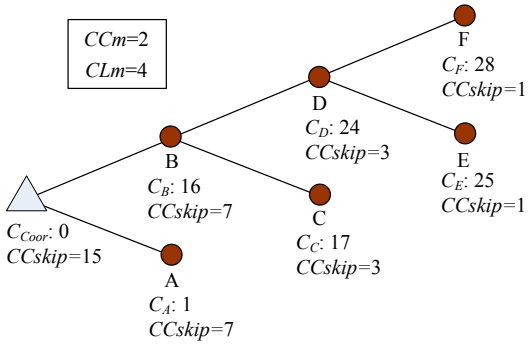


Fig. 6. The logical network of Fig. 5.

3) A cluster head that is not the coordinator should have a link to the bridge of its parent cluster.
4) Conversely, the bridge of a cluster which has child clusters should have a link to the cluster head of each child cluster.
5) A cluster does not cross other clusters and does not have links with other clusters except those locations near-by the cluster head and bridge areas.

After planing the placements, the network manager can construct a *logical network* G_L, in which each cluster is converted into a single node and the parent-child relationships of clusters are converted into edges. For example, Fig. 6 is the logical network of Fig. 5. Then the network manager can determine two parameters: the maximum number of child CCm of a node in G_L and the depth CLm of G_L.

In this work, network addresses are assigned in two stages. With the above planning, the network manager first mutually assigns a cluster ID to each node. Then node IDs are assigned in a distributed manner after network deployment.

The network manager assigns cluster IDs as follows. If $CCm = 1$, there is no branch in the network and the cluster ID assignment is a trivial case. If $CCm \geq 2$, the cluster IDs are assigned following the style of ZigBee in a recursive manner. The nodes in the coordinator's cluster have a cluster ID of 0. For each node at depth d in G_L, if its cluster ID is C, then its i-th child cluster is assigned a cluster ID of $C + (i-1) \times CCskip(d) + 1$, where

$$CCskip(d) = \frac{1 - CCm^{CLm-d}}{1 - CCm}.$$

Fig. 6 shows an example of cluster ID assignment of Fig. 5.

After cluster ID assignment and node deployment, each node periodically broadcasts HELLO packets including its IEEE 64-bit MAC address, 16-bit network address (with its cluster ID but node ID is initiated to *NULL*), and role. Each node maintains a neighbor table to record its neighbors' information. Then the node ID assignment is started by the coordinator broadcasting its beacon with its node ID setting to 0. When a node without a node ID receives a beacon, it will send an *Association_Request* to the beacon sender. If there are multiple beacons, the node with the strongest signal strength will be selected. When the beacon sender, say, v with a network address (C_v, N_v), receives the association request(s), it will do the following steps.

1) If v is *not* a bridge node, it sets a parameter $N = N_v + 1$. Then it sorts request senders according to the received signal strength of their request packets in an descending order into a list L. Then v sequentially examines each node $u \in L$ by the following rules:
   - If $C_u \neq C_v$, v skips u and continues to examine the next node in L.
   - Otherwise, v assigns $N_u = N$ and increments N by 1. Then v replies an *Association_Response* to u with its address. If L is not empty, v loops back and continues to examine the next node in L.

   After finishing the above iteration, v further selects a node u, from the accepted ones, using the following rules: i) If there is a bridge node in the accepted ones, v selects the bridge node. ii) Otherwise, v selects the last node in L. Then v delegates u as the next beacon sender by sending a command *next_beacon_sender*(u) to u.
2) If v is a bridge node, it only accepts the requests from cluster heads of its child cluster. When deciding to accept a node u, v replies an *Association_Response* to u with $N_u = 0$ and a *next_beacon_sender*(u).

For each node u that receives a *next_beacon_sender*(u) in the above steps, it will use the MLME-START primitive defined in IEEE 802.15.4 to start its beacons. Then the same procedure is repeated. Note that we allow a beacon sender to accept multiple children so as to reduce the communication cost of address assignment. Fig. 5 shows some assignment results.

We also design a routing protocol that can utilize nodes' network addresses to facilitate routing. In addition, our routing protocol can take advantage of shortcuts for better efficiency, so our scheme does not restrict nodes to relay packets only to their parent or child nodes as ZigBee does. More details can be found in [12].

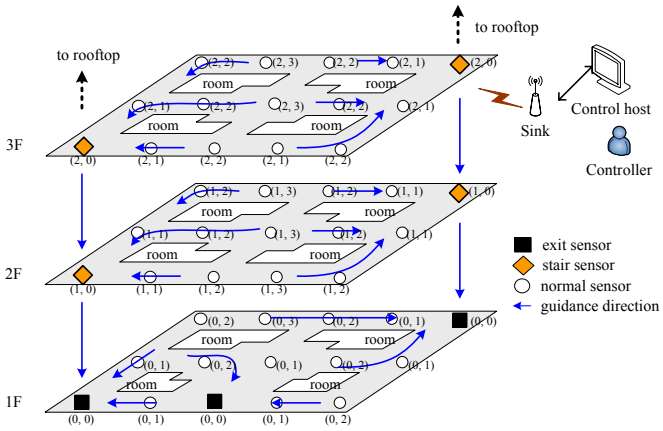


Fig. 7. System architecture of our indoor 3D sensor network.

VI. Dynamically Path Finding for Emergency Guiding in ZigBee-Based WSNs

In this paper, we design a dynamic path finding in a 3D indoor environment for emergency guiding service. Fig. 7 shows the system architecture. Sensor nodes are deployed floor by floor and are connected together by those at stairs. These sensors form a multi-hop ad hoc network. One node serves as the *sink* of the network, and it is connected to the *control host*, which can issue commands and configure the network. From the network, we will construct a *communication graph* $G_c = (V, E_c)$ and a *guidance graph* $G_g = (V, E_g)$, where V is the set of sensors. Each edge $(u, v) \in E_c$ represents a communication link between u and $v \in V$, while each edge $(u, v) \in E_g$ represents a walking path between u and v. Note that a walking path is a physical route that human can pass. So E_g has to be constructed manually based on the floor plane of the building.

After deploying sensors, network initialization starts to construct a reporting tree rooted at the sink for reporting purpose. Then, guidance initialization finds escape paths leading to exits at normal times on the graph G_g. During the guidance initialization, we will compute for each sensor x a weight $w_x = (l_x, alt_x)$, where level l_x is the number of floors from x's current floor to the ground level and altitude alt_x is the guidance hop distance from x to the nearest stair or exit on the same floor. For example, Fig. 7 shows the initialization result of the given deployment.

We design a distributed path finding algorithm for emergency guidance, and allows multiple emergency events and multiple exits coexisting in the sensing field. Our design focuses on quick convergence and can avoid guiding people through hazards. When a sensor detects an emergency event, this sensor and the sensors surrounding it will form a *hazardous region* by raising their weights. Each sensor locally chooses a neighbor with the smallest weight than its weight as the guidance direction. A sensor becomes a local minimum one if it can not find any neighbor with a weight smaller than its own. We use the *partial link reversal* concept to solve this problem. After leading people to stairs, stair sensors will direct people to go downstairs if there is no hazardous downstairs. Otherwise, stair sensors will force people to go to other stairs. If there is no suitable ways to go downstairs, our protocol will guide people to the rooftop to wait for help. More details can be found in [13].

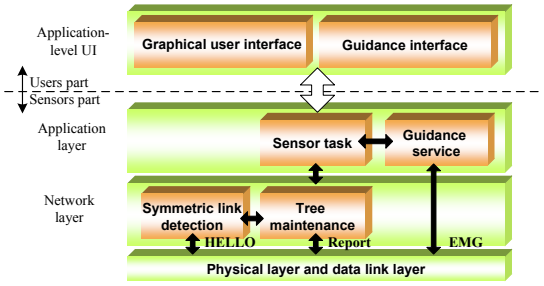


Fig. 8. Protocol stacks of our system.

We implement our system using ZigBee compatible sensor nodes [4] on TinyOS. Fig. 8 shows the protocol stack of our system. Our system can be divided into a "user part" and a "sensor part". In the user part, we implement two application-level user interfaces:

- Graphical user interface (GUI): We provide a JAVA interface for controller to deploy the sensor network, to initialize the network, and to query the statuses of sensors.
- Guidance interface: We implement a LED guidance panel to display the guidance results of sensors. The LED panel is attached to the HiRose connector of the MTS310. When a sensor determines its guidance direction, it will send a control signal to the panel. In our current implementation, this panel can show up to six guidance directions.

For the "sensor part", the protocol is divided into an application layer and a network layer. The former has two components.

- Guidance service: This component implements the guidance protocol to find safe escaping paths.
- Sensor task: At normal time, sensors periodically sense environmental data and report to the control host. When a sensor detects an emergency event, this component will trigger the guidance service component.

The network layer has two components:

- Symmetric link detection: This component measures the quality of communication links between sensors. Sensors periodically send *HELLO* packets. If a sensor finds a neighbor's signal quality is below a threshold for a period of time, this neighbor will be removed from its neighbor table. Only symmetric links will be included into G_c.
- Tree maintenance: This component maintains a sensor's parent. If a sensor's parent is removed from its neighbor table, it will disassociate its descendants and locally find a replacement with the smallest hop count to sink.

We have implemented our emergency guiding system in a virtual building as shown in Fig. 9. Fig. 10 shows two guidance results by placing 27 nodes in a virtual 3-store building. Each

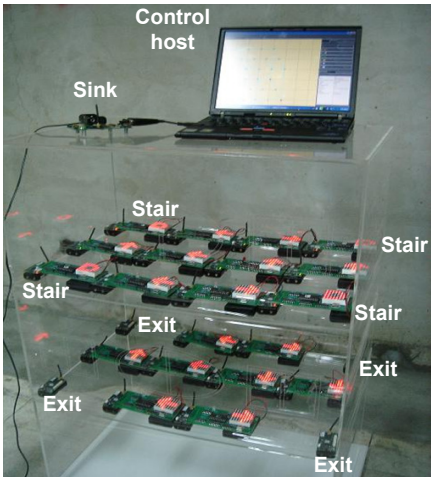


Fig. 9. A virtual building.

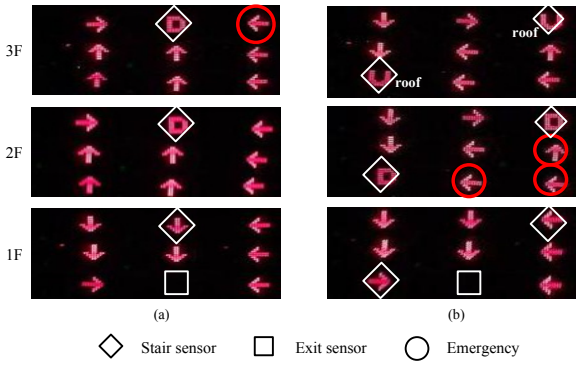


Fig. 10. Guidance results in a 3-store building.

floor is a 3×3 grid network. Fig. 10(a) shows a case that the sensor near the stair sensor on the third floor sensed an emergency event. Since there is no way to roof, sensors will guide people to the exit on the ground floor. In Fig. 10(b), three emergencies are detected. Since the second floor is almost all covered by emergencies, people on the third floor will be guided to the roof. Simulation results for large scale networks can be found in [13].

VII. Conclusions

In this paper, we have introduced four research topics in ZigBee WSNs. First, we discuss the orphan problem in ZigBee-based WSNs. We model this problem by two subproblems, the bounded-delay-and-depth tree formation (BDDTF) problem and the end-device maximum matching (EDMM) problem. The BDDTF problem considers only router-capable devices and is NP-complete, and the EDMM problem considers end devices. Based on the problem formulation, we propose a centralized network formation algorithm. Second, we define a minimum delay beacon scheduling (MDBS) problem for convergecast with the restrictions that the beacon scheduling is compliant to the ZigBee standard. We prove the MDBS problem is NP-complete and introduce centralized and distributed heuristic algorithms. Third, we propose an address assignment scheme for ZigBee-based LT WSNs. The proposed scheme divides nodes into several clusters and then assigns each node a cluster ID and a node ID as its network address. Finally, we introduce our emergency guiding system in a 3D environment based on ZigBee WSNs. The proposed emergency guidance scheme can quickly converge and find safe guidance paths to exits when emergencies occur. Prototyping results show that our protocols can react to emergencies quickly at low message cost and can find safe paths to exits.

Acknowledgements

Y.-C. Tseng's research is co-sponsored by Taiwan MoE ATU Program, by NSC grants 93-2752-E-007-001-PAE, 96-2623-7-009-002-ET, 95-2221-E-009-058-MY3, 95-2221-E-009-060-MY3, 95-2219-E-009-007, 95-2218-E-009-209, and 94-2219-E-007-009, by Realtek Semiconductor Corp., by MOEA under grant number 94-EC-17-A-04-S1-044, by ITRI, Taiwan, by Microsoft Corp., and by Intel Corp.

References

[1] Dust network Inc. http://dust-inc.com/flash-index.shtml.
[2] Design and construction of a wildfire instrumentation system using networked sensors. http://firebug.sourceforge.net/.
[3] Habitat monitoring on great duck island. http://www.greatduckisland.net/technology.php.
[4] Motes, smart dust sensors, wireless sensor networks. http://www.xbow.com/.
[5] H. Choi, J. Wang, and E. A. Hughes. Scheduling for information gathering on sensor network. *ACM/Kluwer Wireless Networks*, 2007, in press.
[6] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 2001.
[7] S. Gandham, Y. Zhang, and Q. Huang. Distributed minimal time convergecast scheduling in wireless sensor networks. In *Proc. of IEEE Int'l Conference on Distributed Computing Systems (ICDCS)*, Lisboa, Portugal, 2006.
[8] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
[9] 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.
[10] 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)(revision of IEEE Std 802.15.4-2003), 2006.
[11] C.-Y. Lin, W.-C. Peng, and Y.-C. Tseng. Efficient in-network moving object tracking in wireless sensor networks. *IEEE Trans. Mobile Computing*, 5(8):1044–1056, 2006.
[12] M.-S. Pan, H.-W. Fang, Y.-C. Liu, and Y.-C. Tseng. Address assignment and routing schemes for zigbee-based long-thin wireless sensor networks. In *Proc. of IEEE Int'l Conference on Vehicular Technology Conference (VTC)*, 2008.
[13] M.-S. Pan, C.-H. Tsai, and Y.-C. Tseng. Emergency guiding and monitoring applications in indoor 3d environments by wireless sensor networks. *Int'l Journal of Sensor Networks (IJSnet)*, 1(1/2):2–10, 2006.
[14] M.-S. Pan and Y.-C. Tseng. The orphan problem in zigbee-based wireless sensor networks. In *Proc. of ACM/IEEE Int'l Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, 2007.
[15] Y.-C. Tseng and M.-S. Pan. Quick covergecast in ZigBee/IEEE 802.15.4 tree-based wireless sensor networks. In *Proc. of ACM Int'l Workshop on Mobility Management and Wireless Access (MobiWAC)*, 2006.
[16] D. B. West. *Introduction to Graph Theory*. Prentice Hall, 2001.
[17] ZigBee specification version 2006, ZigBee document 064112, 2006.