
Chia-Hung Tsai and Yu-Chee Tseng, Fellow, IEEE

Abstract—Although Wireless Sensor Networks (WSNs) have been extensively researched, its deployment is still a main concern. We observe that many monitoring applications for WSNs have adopted a path-connected-cluster (PCC) topology, where regions to be monitored are deployed with clusters of sensor nodes. Since these clusters might be physically separated, paths of sensor nodes are used to connect them together. We call such networks PCC-WSNs. PCC-WSNs may be widely applied in real situations, such as bridge-connected islands, street-connected buildings, and pipe-connected ponds. In this work, we show that the address assignment scheme defined by ZigBee will perform poorly in terms of address utilization. We then propose a systematical solution, which includes network formation, automatic address assignment, and light-weight routing. Simulation results verify the effectiveness of the proposed solution.

Index Terms—Address assignment, protocol designs, routing, wireless sensor networks, ZigBee.

I. INTRODUCTION

A wireless sensor networks (WSN) usually needs to configure itself automatically and support ad hoc routing. A lot of research works have been dedicated to WSNs, including power management [20], routing [12], data gathering [6] [7], sensor deployment and coverage issues [5], and localization [1]. On the application side, habitat monitoring is explored in [16], wildfire monitoring is addressed in [9], healthcare system is proposed in [3], and navigation is studied in [24].

To form a WSN, two most important issues are addressing and routing. Strict per-node addressing is expensive in a dense network, because not only would the address space be large, but also these addresses would need to be allocated and managed according to the topology change. Allocation of addresses in a dense network is a problem which is often underestimated [14]. On the other hand, routing is to discover paths from source nodes to destination nodes based on their network addresses. Path discovery in a dense network could incur high communication overhands. Therefore, designing a light-weight addressing and routing protocol for WSNs is very important.

Recently, ZigBee [28] has been proposed for addressing and routing on WSNs. It supports three kinds of network topologies, namely star, tree, and mesh networks. A ZigBee coordinator is responsible for initializing, maintaining, and controlling the network. Star networks can only cover small areas. For tree and mesh networks, communications can be conducted in a multi-hop fashion. The backbone of a tree/mesh network is formed by one ZigBee coordinator and multiple ZigBee routers. An end device must associate with the coordinator or a router. In a tree network, routing can be done in a stateless manner; a node can simply route packets based on nodes’ 16-bit short addresses, which are assigned based on the tree structure. In fact, a mesh network also has a tree inside to serve as its backbone; routing can go directly along the tree without route discovery or go along better paths if a node is willing to conduct route discovery first.

In the literature, most works have assumed that a ZigBee network grows in an arbitrary manner. Recently, the long-thin topology (Fig. 1(a)) has been proposed for applications where sensor deployment is subject to environmental constraints [18]. The use of long-thin network ranges from leakage detection of fuel pipes [10] [22] [26], tunnel monitoring, street lights monitoring [11], flood protection of rivers [18], debris flow monitoring [15], barrier coverage [4], and in-sewer gas monitoring [13]. In this paper, we further extend the long-thin topology to a path-connected-cluster (PCC) topology, where regions requiring intensive sensing are deployed with clusters of sensor nodes and these clusters, which are physically separated, are connected by long paths for occasional communications. We call such topologies PCC-WSNs. Fig. 1(b) shows an application for habitat monitoring in a wildlife park. Sensors for different habitat zones form different clusters. Data from these clusters is collected through paths connecting them. Such “sometimes fat, sometimes slim” topologies would worsen the orphan problem [19], which states that the ZigBee address assignment may not allow some nodes (called orphans) to join the network even if there are available addresses elsewhere (refer to Section II-B for more discussions).

Although ZigBee supports address-based routing through its distributed addressing scheme, it could incur a lot of orphans or result in waste of address space [19]. The virtual coordinate addressing schemes in [8] try to provide stateless routings directly from nodes’ addresses. However, additional GPS devices or localization mechanisms should be involved. Moreover, these schemes still need a lot of address spaces. Address assignments for WSNs are studied in [2], [14], [17], [21], [27]. These works all focus on compact assignment of addresses to sensor nodes, but they need additional routing protocols to deliver packets because they do not support address-based routing. The work [14] allows network addresses to be reused to conserve address space, but it only supports many-to-one
communication.

The goal of our work is to propose an address-light and routing-light protocol for PCC-WSNs. Our approach is based on the principle of ZigBee address assignment, but leads to much more compact address usage than the original ZigBee’s design, thus significantly alleviating the orphan problem in PCC-WSNs. Furthermore, based on our addressing, routing still incurs low communication overheads. This work contributes in formally defining the PCC-WSN topology. Given a PCC-WSN, we present a formation scheme to automatically separate paths from clusters in a distributed manner. Then we propose a ZigBee-like address assignment scheme for a PCC-WSN. In particular, we design different addressing strategies for slim parts (paths) and fat parts (clusters) of a PCC-WSN. This design allows us to conduct light-weight, address-based routing. Although this requires slight modification to ZigBee specification, we find this leads to quite efficient communications. The rest of this paper is organized as follows.

Section II gives some preliminaries. Section III presents our algorithms, including formation, addressing, and routing of a PCC-WSN. Some discussions for further generalizing our network formation are in Section IV. Section V presents our performance evaluation results. Section VI concludes this paper.

II. PRELIMINARIES AND PROBLEM DEFINITION

A. ZigBee Address Assignment and Tree Routing

In ZigBee, network addresses are assigned to devices in a distributed manner. To form a network, the coordinator determines the maximum number of children per router \((C_m)\), the maximum number of child routers per router \((R_m)\), and the maximum depth of the network \((L_m)\). Note that children of a router include child routers and child end devices. So \(C_m \geq R_m\) and up to \(C_m - R_m\) children of a router must be end devices (an end device cannot have children). Addresses are assigned in a top-down manner. The coordinator takes 0 as its address and divides the remaining address space into \(R_m + 1\) blocks. The first \(R_m\) blocks are to be assigned to its child routers and the last block has \(C_m - R_m\) addresses, each to be assigned to a child end device. The similar approach is adopted by each router to partition its given address space. From \(C_m, R_m, \) and \(L_m\), each router at depth \(d\) can compute a \(C_{skip}(d)\) value, which is the size of one address block to be assigned to a child router:

\[
C_{skip}(d) = \begin{cases} 
1 + C_m \times (L_m - d - 1), & \text{if } R_m = 1 \\
1 + C_m - R_m - C_mR_mL_m - d - 1, & \text{otherwise}.
\end{cases}
\]

The value of \(d\) is 0 for the coordinator and is increased by one after each level. For example, given an address block, a router at depth \(d\) will take the first address for itself, reserve \(R_m\) blocks, each with \(C_{skip}(d)\) addresses, for its child routers, and reserve \(C_m - R_m\) addresses for its child end devices. Fig. 2 shows an example of ZigBee address assignment. Clearly, in Fig. 2, the value of \(R_m\) is at least 3 for supporting 3 router children. Note that ZigBee network address is 16 bits. Even though we set \(L_m\) to 9, \(B\) and \(C\) still cannot associate with the network. Even worse, such address assignment would work poorly in a PCC-WSN because of its “sometimes fat, sometimes slim” nature.

With the above address assignment, ZigBee supports very simple address-based routing. When a router receives a packet for \(A_{dest}\), it first checks if it is the destination or one of its children is the destination. If so, it accepts the packet.
or forwards this packet to its child whose address block contains $A_{\text{dest}}$. Otherwise, it relays the packet to its parent. Assume that the depth of this router is $d$ and its address is $A$. This packet is forwarded to its child $A_r$ which satisfies $A_r < A_{\text{dest}} < A_r + C_{\text{skip}}(d - 1)$ such that $A_r = A + 1 + \left\lfloor \frac{A_{\text{dest}} - (A + 1)}{C_{\text{skip}}(d)} \right\rfloor \times C_{\text{skip}}(d)$. If the $A_{\text{dest}}$ is not a descendant of $A$, this packet will be forwarded to its parent. Note that in a mesh network, nodes are also assigned addresses following these rules. This means that address-based routing can coexist with a mesh routing.

B. Definition of PCC-WSN

A WSN is modeled as an undirected graph $\mathcal{G} = (V, E)$, where $V$ contains all nodes and $E$ contains all communication links between nodes. Each edge in $E$ is bidirectional (we do not consider directed links). One special node $t \in V$ is designated as the coordinator. A PCC-WSN has a special topology in that $V$ can be divided into two sets $C$ and $P$, where $C$ is a set of clusters and $P$ is a set of paths. A cluster in $C$ is a group of connected nodes with dense connectivity. A path in $P$ is a linear topology with at least $\delta$ nodes and each node has a degree $\leq 2$. An end node of a path either connects to a cluster node or is a terminator itself (which has degree $= 1$). Intuitively, we assume that clusters are sufficiently separated physically by at least $\delta$ nodes. Therefore, there may exist short paths (with length $\leq \delta$) in a cluster. The value of $\delta$ can be chosen by the network administrator according to physical constraints. Fig. 3 shows an abstraction of PCC-WSN. A possible instance of Fig. 3(a) is shown in Fig. 3(b). Here, $\delta = 5$, so nodes in $H$ form a path. (Note that node $w$ in $H$ can be a degree-1 terminator or degree-2 node connecting to another cluster.) Nodes $x$, $y$, and $z$ in $C$ are not regarded as a path, but as a part of cluster $C$.

III. NETWORK FORMATION, ADDRESSING, AND ROUTING PROTOCOLS

Given a PCC-WSN, we propose a low-cost, fully automated scheme to initialize it, assign addresses to nodes, and conduct ZigBee-like tree routing. First, a distributed network formation procedure will be launched by the coordinator $t$ to divided nodes into two sets $C$ and $P$. Then, a two-level address assignment scheme is conducted to assign a level-1 and a level-2 addresses to each node. A level-1 address is to uniquely identify a path or a cluster. A level-2 address is similar to ZigBee addressing but is confined within one cluster/path. For simplicity, we assume that all nodes are router-capable devices. Finally, we show how to conduct routing based on our two-level addressing. Also, we address how our protocol can adapt to changeable topologies.

A. Network Formation

Given a PCC-WSN $\mathcal{G} = (V, E)$, the network formation process has three goals: (i) to partition $\mathcal{G}$ into clusters and paths, (ii) to assign a group ID (GID) to each cluster/path, which should be known to each member in that cluster/path, and (iii) to identify an entry node for each cluster/path, which is the one nearest to $t$ in terms of the number of clusters/paths if we travel from $t$ to the entry node (as a special case, $t$ will serve as the entry node of its cluster). At this stage, addressing is based on devices’ MAC addresses. It is assumed each node $v$ has a unique MAC address $MAC(v)$. The GID of a cluster/path will be the MAC address of its entry node. For example, in Fig. 3, each cluster/path has an entry node. Nodes $x_3$ and $x_5$ are the entry nodes of path $H$ and cluster $C$, respectively.

In our protocol, a node has four states, as shown in Fig. 4. First, it is in the INIT (initial) state. In the CLF (classification) state, it tries to decide if it is within a cluster or a path. In the PB (probe) state, it helps to determine the range of its cluster/path and elect the entry node of its cluster/path. Finally, it enters the TERM (terminated) state. The detailed process is discussed below.

1) Initially, all nodes are in the INIT state. The coordinator $t$ first enters the CLF state and starts the network formation process by broadcasting a START message around the network. Each node receiving a START for the first time should help rebroadcast it.

2) On a node $v$ receiving the first START message, it enters the CLF state. Then $v$ tries to determine its degree (i.e., number of neighbors) in $\mathcal{G}$. This can be easily achieved by nodes exchanging periodically HELLO packets. According to its degree, $v$ classifies its type by setting $type(v) = \text{“cluster”}$ if its degree is $\geq 3$ and setting
type(v) = “t-path” otherwise. Here, “t-path” means a tentative path.

3) If v has type(v) = “t-path”, it needs to confirm its type further (recall that a path must contain at least δ nodes). To do so, if v is adjacent to a node of type “cluster” (i.e., it is an end node of a tentative path), it sends a TRAVERSE packet to calculate the length of its path. (This can be done by sending a TRAVERSE packet containing len = 1 to its neighbor with type “t-path”. On u receiving the packet and v does not have a neighbor with type “cluster”, it sets len = len + 1 and forwards the packet to the other direction. Otherwise, u is the end of the path. It checks if (len + 1) ≥ δ. If so, u notifies all nodes along the reverse path to change their types to “path”; otherwise, u notifies them to change their types to “cluster”.)

4) A node v, once confirming its type as “cluster” or “path”, will enter the PB state. The probe process involves two messages, C-PROBE and P-PROBE, for searching the ranges of clusters and paths, respectively. v keeps a variable GID(v) to track its group ID, a variable Dist(v) to track its distance to the coordinator, in terms of the number of clusters/paths from t to v’s cluster/path, and a variable PAR(v) to track its parent cluster/path if v is an entry node. Initially, v sets all variables to ∞, except the coordinator t, which sets GID(t) = MAC(t), Dist(t) = 0, and PAR(t) = MAC(t).

This process is started by t after it enters the PB state. It first broadcasts a C-PROBE(GID(t), Dist(t)). Below, given two pairs (GID, Dist) and (GID’, Dist’), we say that (GID, Dist) < (GID’, Dist’) if (Dist < Dist’) or (Dist = Dist’ and GID < GID’). C-PROBE and P-PROBE are propagated by the following rules.

- On v of type(v) = “cluster” receiving a C-PROBE(g, d), it checks if (g, d) < (GID(v), Dist(v)). If so, it updates its GID(v) = g and Dist(v) = d and broadcasts a C-PROBE(GID(v), Dist(v)).
- On v of type(v) = “path” receiving a P-PROBE(g, d), it checks if (g, d) < (GID(v), Dist(v)). If so, it updates its GID(v) = g and Dist(v) = d and broadcasts a P-PROBE(GID(v), Dist(v)).
- On v of type(v) = “cluster” receiving a P-PROBE(g, d), it checks if (MAC(v), d + 1) < (GID(v), Dist(v)). If v will bid to serve as the entry node of its cluster and update its GID(v) = MAC(v), Dist(v) = d + 1, and PAR(v) = g. Then v broadcasts a C-PROBE(GID(v), Dist(v)).
- On v of type(v) = “path” receiving a C-PROBE(g, d), it checks if (MAC(v), d + 1) < (GID(v), Dist(v)). If so, v will bid to serve as the entry node of its path and update its GID(v) = MAC(v), Dist(v) = d + 1, and PAR(v) = g. Then v broadcasts a P-PROBE(GID(v), Dist(v)).

5) After all nodes are stable, they enter the TERM state.

We make some remarks below. The above process tries to form a minimum spanning tree by regarding each path/cluster as a super-node. Each node which is in a cluster and connects to a path could be a candidate entry node. Note that every node in a path has degree ≤ 2. By our definition, an end point in a path will connect to only one node in a cluster. For example, in Fig. 3(a), cluster C could have four candidate entry nodes, and the one which has the shortest distance to the coordinator will become the entry node (in this example, it is x_v). In Fig. 3(b), x_5 is the only node connected to H. If there exist multiple candidate entry nodes having the same distance to the coordinator, we use nodes’ MAC addresses to break the tie. The selection of an entry node of a path is similar. Hence, there exists only one entry node in each supernode. Also, a node v knows that it is an entry node if MAC(v) = GID(v). Also note that nodes may not enter the same state at the same time. For example, a node in the CLF state may receive a “premature” C-PROBE/P-PROBE message, which it is unable to process yet. In this case, the receiving node will buffer such packets. Once it enters the legal state, it can retrieve them for processing. For simplicity, we did not discuss how we know that all nodes are “stable” in step 5. This can be achieved by distributed termination detection protocols [25], and we omit the details.

We further discuss the effect of δ. A smaller δ could result in many short paths connecting clusters. This may increase the number of supernodes in our level-1 addressing. Thus, larger Cm(1) and Lm(1) may be needed, resulting in larger address spaces. But the impact on lever-2 address space is limited. Reversely, a larger δ may prohibit some “paths” from becoming paths. Thus, there may be less supernodes leading to smaller Cm(1) and Lm(1). However, combining clusters and paths requires larger Lm(2). Also, increasing Lm(2) by one results in doubling the level-2 address space. Hence, making δ just a little bit smaller than the average length of paths is preferred.

B. Address Assignment

We propose a two-level addressing. It has two purposes: (i) to reduce address space and (ii) to support ZigBee-like stateless routing. In level-1 addressing, we regard each cluster/path as a supernode and use ZigBee-like addressing to assign an n-bit address to each supernode. In level-2 addressing, we again apply the ZigBee-like addressing on each individual cluster/path to assign an n-bit address to each node. The concatenation of the level-1 and the level-2 addresses forms a node v’s network address, denoted by (L_1(v), L_2(v)). During this process, we will also construct a Descendant Table (DT), which allows an entry node to reach the entry nodes of its child supernodes.
Before executing our two-level addressing, we need to determine \( m \) and \( n \) first. It relies on forming a logical tree from all supernodes and forming a BFS tree from all nodes of each supernode. We summarize the steps as follows.

**Step 1:** For each node \( v \) entering the TERM state and serving as an entry node, it reports the pair \((\text{PAR}(v), \text{GID}(v))\) to \( t \). From these information, \( t \) constructs a logical tree \( T_L \) of supernodes. (For example, Fig. 5 is the logical network of Fig. 3.) With \( T_L \), \( t \) determines its tree parameters as \( Cm^{(1)} = Rm^{(1)} = (\text{max child degree of } T_L) \) and \( Lm^{(1)} = (\text{height of } T_L) \). This would require \( M = \lceil \log_2 M \rceil \) bits for addressing, where \( M = \lceil \frac{Cm^{(1)}Lm^{(1)}}{2} \rceil = \lceil \frac{Cm^{(1)}-1}{2} \rceil \) is the maximal number of nodes in \( T_L \). (In the example of Fig. 5, \( Cm^{(1)} = Rm^{(1)} = 3 \) and \( Lm^{(1)} = 4 \).)

**Step 2:** For each node \( v \) entering the TERM state and serving as an entry node, it forms a BFS tree \( T \) among its members. With \( T, v \) determines its tree parameters as \( Cm^{(2)}(v) = Rm^{(2)}(v) = (\text{max child degree of } T) \) and \( Lm^{(2)}(v) = (\text{height of } T) \). This would require \( N(v) = \lceil \log_2 N \rceil \) bits for addressing, where \( N = \lceil \frac{Cm^{(2)}(v)Rm^{(2)}(v)+1}{2} \rceil \) is the maximal number of nodes in \( T \). Each entry node \( v \) reports its \( n(v) \) to \( t \). Then \( t \) chooses the maximum one from all \( n(v) \)'s as the value of \( n \).

The above process has determined \( m, n \), and the tree formation parameters for level-1 and level-2 addressing. Next, \( t \) can start our two-level address assignment. It sets \( L_1(t) = L_2(t) = 0 \) and periodically broadcasts beacons containing \( m, n, L_1(t), L_2(t) \), the global level-1 parameters \( (Cm^{(1)}, Rm^{(1)}, Lm^{(1)}) \), and its level-2 parameters \( (Cm^{(2)}(t), Rm^{(2)}(t), Lm^{(2)}(t)) \). Our address assignment follows the ZigBee style recursively, but in a two-level manner. When a node \( u \) without a network address receives a beacon, it will send an Association Request to the beacon sender. If it receives multiple beacons, the node with the strongest signal strength will be selected. When the beacon sender, say \( v \), receives the association request, there are two cases:

**Case 1:** \( u \) and \( v \) belonging to different supernodes. If \( u \) is an entry node, \( v \) will process the request. \( v \) first unicasts the request packet to its entry node, say \( e \), to retrieve a level-1 \( m \)-bit address for \( u \). (Note that this unicast scheme will follow our routing protocol described in Section III-C.) If this is the \( i \)-th request received by \( e \), it will respond the following level-1 address to \( v \).

\[
L_1(u) = L_1(v) + (i - 1) \times Cskip^{(1)}(d) + 1,
\]

where \( Cskip^{(1)}(d) = \frac{1-Cm^{(1)}Lm^{(1)}}{1-Cm^{(1)}Lm^{(1)}} \). Then \( v \) replay an Association Response with an address \((L_1(u), 0)\) to \( u \). Note that for making our routing simple and efficient, the entry node \( e \) will also memorize this information \((L_1(u), L_2(v))\) into its DT. The detailed functionality of the DT table will be addressed in Section III-C. On receipt of \( v \)'s response, \( u \) will start sending beacons containing \( m, n, L_1(u), L_2(u) \), the global level-1 parameters, and its own level-2 parameters \((Cm^{(2)}(u), Rm^{(2)}(u), Lm^{(2)}(u))\).

**Case 2:** \( u \) and \( v \) belonging to the same supernode. If this is the \( i \)-th request received by \( v \), it will respond the following level-2 address to \( u \).

\[
L_2(u) = L_2(v) + (j - 1) \times Cskip^{(2)}(d) + 1,
\]

where \( Cskip^{(2)}(d) = \frac{1-Cm^{(2)}(v)Rm^{(2)}(v)-d}{1-Cm^{(2)}(v)Rm^{(2)}(v)} \). On receipt of \( v \)'s response, \( u \) will update its \( L_2(u) \) and set \( L_1(u) = L_1(v) \). Then \( u \) will start sending beacons similar to case 1.

In Fig. 6, we have shown a concrete subgraph of Fig. 3 with some assignment results, where each address is expressed in Hex and the first two symbols represent the \( m \)-bit address and the last two represent the \( n \)-bit address. Through Fig. 5, \( Cm^{(1)} = Rm^{(1)} \) and \( Lm^{(1)} \) can be determined as 3 and 4, respectively. Hence, \( m = 7 \) bits and the \( L_1(x_2) = 0 + \frac{8}{3} \times 1 + 1 = 41 \), where \( x_2 \) is the second child of \( t \). Also, \( n \) is determined according to \( n(x_{10}) \), whose \( Cm^{(2)} = Rm^{(2)} = 5 \) and \( Lm^{(2)} = 3 \), which is conducting the largest level-2 address space. Therefore, \( n = 8 \) bits and the level-2 address of the second child of \( x_{10} \) is \( 0 + \frac{8}{3} \times 1 + 1 = 32 \). Other more addressing results will be given in Fig. 6. Note that our two-level addressing has better address space utilization than the pure ZigBee address assignment because each cluster/path can have its own network parameters.

**C. Routing**

With our two-level addressing, we also design a two-level routing approach consisting of a level-1 routing and a level-2 routing. The former can be imagined as routing in \( G_L \), which can assist in routing packets to the supernode containing the destination. The later is to route packets simply within the supernode (the same cluster/path). Therefore, suppose that a node \( v \) receives a packet destined to \( dst \). If \( L_1(dst) = L_1(v) \), \( v \) can simply adopt the level-2 routing to transmit packets to \( dst \). Otherwise, \( v \) will first perform the level-1 routing until \( L_1(dst) = L_1(v) \). Then \( v \) also applies the level-2 routing to transmit packets to \( dst \). Note that the concept of our level-1 routing is to determine which cluster/path the packets should be forwarded to and how to forward the packet to that cluster/path. When routing to cross the cluster/path, it still applies the level-2 routing.

Based on our two-level addressing, given a source \( x \) and a destination \( y \) both in the same cluster/path, the distance \( P^{(2)}(L_2(x), L_2(y)) \) between them can be easily determined.
as follows:

\[ P^{(2)}(L_2(x), L_2(y)) = D^{(2)}(L_2(x)) + D^{(2)}(L_2(y)) - 2 \times D^{(2)}(LCA^{(2)}(L_2(x), L_2(y))), \]

(4)

where \( D^{(2)}(L_2(x)) \) is the depth of node \( x \) to its entry node, and \( LCA^{(2)}(L_2(x), L_2(y)) \) is the least common ancestor’s level-2 address of \( x \) and \( y \). Also, according to [23], our addressing can further make our routing take shortcuts to improve the routing performance. Hence, after obtaining a network address, each node should periodically broadcast HELLO packets including its network address. In this work, we consider only symmetric links. A communication link \((u, v)\) is established only if \( u \) receives \( v \)'s HELLO including \( u \) as its neighbor and the HELLO’s signal quality is above a threshold. Note that the signal quality should be the average of several packets. Then when routing a packet, each node will transmit the packet to a neighbor which has shortest tree path to the destination, which is determined by Eq. (4).

Moreover, for making our routing more efficient, we further introduce Descendant Table (DT) table. The DT is introduced for enhancing our level-1 routing. Although our level-1 addressing is still based on ZigBee style address assignment, each entry node not only cannot directly communicate with its descendant entry node, but also has no any routing information to its descendant entry node. Hence, our addressing makes each entry node maintain a DT which is discussed and generated in Section III-B. Then each entry node will be aware of by through which member node it can communicate with its descendant entry nodes. Remind that each DT entry has the following format: \((L_1(x), L_2(y))\), which stands for the entry node \( x \) obtaining its network address through node \( y \). Then the entry node will know it can communicate with its descendant node \( x \) through \( y \).

Hence, suppose that \( v \) receives a data packet or has a data packet destined to \( dst \). Our two-level routing is performed as follows.

1) If \( dst \) is a neighbor of \( v \), \( v \) will send this packet directly and terminate the process.

2) If \( L_1(dst) = L_1(v) \), \( v \) determines a neighbor \( u \) such that \( L_1(u) = L_1(dst) \) and \( P^{(2)}(L_2(u), L_2(dst)) \) is minimized. Then \( v \) will forward this packet to \( u \).

3) If \( L_1(dst) \neq L_1(v) \), according to the ancestor-descendant relationship between \( v \)'s and \( dst \)'s supernodes, there are two cases.

- **If \( dst \)'s supernode is a descendant of \( v \)'s supernode in \( \mathcal{G}_C \), i.e., \( L_1(v) < L_1(dst) < L_1(v) + Cm^{(1)} \times Cskip^{(1)}(D^{(1)}(L_1(v))) + 1 \), \( v \) will first check if it is serving as an entry node. If yes, \( v \) will find an DT entry \((L_1(des), L_2(p))\) which satisfies \( L_1(des) \leq L_1(dst) < L_1(des) + Cm^{(1)} \times Cskip^{(1)}(D^{(1)}(des)) + 1 \) from its DT. Then \( v \) finds a neighbor \( u \) such that \( P^{(2)}(L_2(u), L_2(p)) \) is minimized, appended this DT entry into this packet, and finally forwards this packet to \( u \). Otherwise, \( v \) first checks if the packet has appended any DT routing entry. If no, \( v \) finds a neighbor \( n \) such that \( P^{(2)}(L_2(n), 0) \) is minimized, and then try to forward this packet to its entry node. Reversely, if this packet has been already appended a DT entry \((L_1(des), L_2(p))\), there are two cases.

  - If \( L_1(v) = L_1(p) \) but \( v \neq p \), \( v \) determines a neighbor \( u \) such that \( P^{(2)}(L_2(u), L_2(p)) \) is minimized. Then \( v \) forwards this packet to \( u \).

  - If \( L_1(v) = L_1(p) \) and \( v = p \), \( v \) simply forwards this packet to its neighbor \( u \) with address \((L_1(des), 0)\).

- **If \( dst \)'s supernode is not a descendant of \( v \)'s supernode in \( \mathcal{G}_C \), i.e., \( L_1(dst) \leq L_1(v) \) or \( L_1(dst) \geq L_1(v) + Cm^{(1)} \times Cskip^{(1)}(D^{(1)}(L_1(v))) + 1 \), \( v \) will first check if it has a neighbor \( anc \) which satisfies \( L_1(anc) < L_1(v) < L_1(anc) + Cm^{(1)} \times Cskip^{(1)}(D^{(1)}(L_1(anc))) + 1 \). If there are multiple candidates, \( v \) will choose the one such that \( P^{(2)}(L_2(anc), 0) \) is minimized. Then \( v \) forwards the packet to \( anc \). Otherwise, \( v \) determines a neighbor \( u \) with minimal \( P^{(2)}(L_2(u), 0) \), and try to forward this packet to its entry node.

Note that the above procedure will be repeated until \( dst \) receives the packet. Our routing design tries to strike a balance between efficiency and simplicity. It basically follows the ZigBee tree-like routing. However, making shortcut on the PCC-WSN is possible due to the existence of neighbor
tables, routing table, and our design of hierarchical network addresses. Therefore, unlike the original ZigBee tree routing, nodes are not restricted to relay packets only to their parents or children.

D. Topology Maintenance

Here, we discuss how our protocol deals with topology changes incurred by new coming nodes, leaving nodes, or dead nodes. We basically follow the specification in IEEE 802.15.4 to handle this issue. A leaving node dissociates with its parent by issuing a NLME-LEAVE request command. The network addresses formerly assigned to it and its children should be released to other nodes that join subsequently. A dead node can be detected by its parent by an aging process. Similarly, the addresses assigned to it and its children should be released. Children of a dead node will reassociate with a new parent by NLME-SYNC-LOSS indication or link failure detection. Then all descendants will perform the reassociation procedures in a recursive manner. On the other side, when a new coming device wants to join the network, it can simply issue a NLME-JOIN request. Then a router which receives the request and still has routing capacities may accept it. In case that a new coming node can not successfully join the network, we can perform address reassignment and re-calculate the network parameters. Note that this can be conducted within a supernode. The routing capacities designed by our addressing scheme are big enough to serve a lot of new coming nodes. Only when it is necessary, an overall restructuring is needed. Therefore, our design can significantly reduce the network address assignment overhead on the topology maintenance as compared to the original ZigBee design.

E. Complexity of Network Formation

Compared to the ZigBee protocol, our design requires three extra packets (START, TRAVERSE, and C-PROBE/P-PROBE) and one additional BFS procedure. Both the START packet and the BFS procedure require flooding the network. Flooding of C-PROBE/P-PROBE is only confined within a cluster or a path. Thus, an efficient broadcast scheme is needed (this is beyond the scope of this paper). Let n be the total number of nodes in the network. The message complexity is $O(n)$. Flooding of the TRAVERSE packet is only confined in a path. In the worst case, the message cost is $O(n)$. Overall, the additional message complexity as opposed to ZigBee is $O(n)$.

Also, the additional message overhead incurred by our design is only incurred at the formation and possible reformation. There are no additional overheads during network operations. So we believe that such extra overheads are worthwhile since not only the orphan problem is relieved, but also the addresses of nodes, which need to be sent in most packets are significantly reduced.

IV. DISCUSSION

Recall that the paths are used to connect physically separated regions requiring intensive sensing. The definition of a path limits the degree of each node to less than 2. However, such paths could easily make the whole network disconnect due to the existence of dead nodes or unstable links. For making our design more practicable, we can loosen the constraint of the degree to fit to more general PCC-WSNs. A path is still a long-thin topology physically, but each node can have more than 2 neighbors for network reliability. Under such network scenario, our network formation will fail on classifying C and P. Hence, we propose a Hand In Hand (H2) protocol to substitute the step 2 and step 3 of our formation protocol to reinforce the path recognition. Also, we do some modifications on our addressing protocol to accommodate with such definition of paths.

The key idea of H2 is the signal strength as weaker as the distance larger. Each node determines two neighbors with strongest link qualities, which we call strongest neighbors. If there exists two nodes both belonging to the other’s strongest neighbors, we link both nodes hand in hand. Finally, all nodes in a path ideally form a line hand in hand by our H2 protocol. The number of neighbors linking hand in hand with a node $v$ is denoted by $HDeg(v)$, substituting for the role of the degree in Section III-B. Clearly, for each node $v$, $HDeg(v) \leq 2$. Our H2 protocol has three steps: i) path formation, ii) path traverse, and iii) path shrinking, and is presented as follows.

(Path formation) On a node $v$ entering the CLF state, it determines its $HDeg(v)$ first. This can be achieved by each node piggybacking the MAC addresses of its strongest neighbors in the periodic HELLOs. If $v$ is included in $u$’s HELLO packets and $u$ belongs to $v$’s strongest neighbors, $v$ sets $HDeg(v) = HDeg(v) + 1$. Then $v$ sets $type(v) =$ “t-path” if $HDeg(v) > 0$. Otherwise, it sets $type(v) =$ “cluster”.

(Path shrinking) If a node $v$ has type “t-path”, it checks if there exists neighbors not connecting to it hand in hand but having type “cluster”. If yes, $v$ sends a SHRINK packet to its neighbor(s) linking hand in hand and sets $type(v) =$ “s-cluster”, which represents as type ”cluster” but shrinks from a tentative path. On each node $u$ with type “t-path” receiving the SHRINK packet, it sets $HDeg(u) = HDeg(u) − 1$. If $HDeg(u) = 0$, $u$ also sets $type(v) =$ “s-cluster”.

(Path traverse) If a node $v$ has $type(v) =$ “t-path”, $HDeg(v) = 1$, and no any neighbors with type “cluster”, it further confirms its type by sending a TRAVERSE packet to its connecting hand in hand neighbor to calculate the length of the path as Section III-A does.

Obviously, after path formation step of our H2 protocol,
there may exist nodes with type “t-path” forming many kinds of shapes shown in Fig. 7, where nodes belonging to shapes A, B, C, and D are all classified as “t-path”. Through our path shrinking, the over-length path A or erroneous paths in judgment B and C can be shrunk due to the existence of neighbors with type “cluster”. Then, relying on our path traverse, remaining misclassified nodes can be corrected back to type “cluster”. However, some close shapes such as D could never been traversed by TRAVERSE packets. Hence, on a node setting its type to “t-path” or “s-cluster”, it fires a timer. If the timer expires, a node still having type “t-path” or “s-cluster” will automatically change its type to “path” or “cluster”, respectively.

Because our H2 protocol enhances the path recognition and revises the original formation scheme, our addressing procedure only needs a minor modification. If both the senders with type “path” of the beacon and the association request are belonging to the same supernode, the sender of the beacon can only process that request originating from one of its hand-in-hand neighbors. Otherwise, the addressing process is unchanged.

V. PERFORMANCE EVALUATIONS

We simulate some PCC-WSNs that are generated by a systematical method which has been developed by C language. An $S \times S$ rectangle region is simulated, on which $n$ sensor nodes are randomly deployed. The field is divided into grids, each size of $s_g \times s_g$ $m^2$. In order to form a PCC-WSN in a systematic way, we impose a fail probability of $P_f$ on each grid. If a grid is determined to fail, all sensor nodes inside the grid fail. This would partition the network into multiple subnetworks when $P_f$ is sufficiently large. The successful and adjacent grids will be grouped into the same cluster by our simulator. Then we apply a minimum spanning tree algorithm to build paths during clusters. Our simulator generates sensor nodes at every distance of $d$ on each path. The coordinator $t$ is at the left-top corner. Hence, we make the left-top grid always be not failed. Fig. 8 shows an example of a random generated PCC-WSN. The default values of all parameters are set as follows: $n = 700$, $S = 700$ $m$, $s_g = 100$ $m$, $P_f = 95\%$, and $d = 30$ $m$. All simulation results are from the average of 100 runs.

As Fig. 8 shows, our protocol can partition the network nodes into 2 sets accurately. Also, each node can successfully connect to the network by our addressing assignment. Based on the same determination principle of $C_m$, $R_m$, and $L_m$, ZigBee addressing can still make all nodes connect to the network as well. However, the address space conducted by ZigBee is excessively larger than that conducted by our protocol. In Fig. 9, the address space is represented by logarithm. Larger transmission range will result in smaller address space. This is because larger transmission range will decrease the value of $L_m$. Moreover, the address space conducted by our protocol is smaller than that conducted by ZigBee addressing up to $10^{100}$.

The address space is mainly influenced by the values of $C_m$ and $L_m$. Below, we will limit the address space. We mainly consider the number of orphans as our performance metrics. Also, we vary some factors such as the number of sensor nodes and the transmission ranges to investigate the performance. First, in our protocol, we fix $C_m$ and $R_m$ from 3 to 6 but keep $L_m$ determined by the original principle unchanged. In ZigBee addressing, we limit the address space as the one determined by our protocol. Therefore, based on $C_m$, $R_m$, and address pool $A$, the $L_m$ of ZigBee can be determined as $L_m = \frac{\log (A \times (C_m - 1) + 1)}{\log C_m} - 1$. Fig. 10(a) shows that our protocol has better performance on orphans against ZigBee. ZigBee has very poor performance due to the existence of paths. Larger $C_m$ will result in fewer orphans in our protocol because it will cause larger address space. However, larger transmission range could incur more orphans. This is because larger transmission range could increase the probability of no routing capacity for a router. Moreover, Fig. 10(b) shows that our protocol has better performance on orphans against ZigBee. ZigBee addressing, we limit the address pool as the one determined by our protocol. Therefore, we vary $C_m$ and $L_m$ of ZigBee to measure the orphans. In Fig. 11, we vary $C_m$ of ZigBee from 3 to 6. Clearly, our protocol does not incur any orphans as Fig. 11(a) shows. However, ZigBee addressing will result in great part of nodes as orphans. As the network nodes increase, orphans will also increase. Fig. 11(b) shows the corresponding $L_m$ of ZigBee on varying the $C_m$ and fixing the address space. Obviously, ZigBee addressing incurs poor
For reducing the influence of the paths on ZigBee addressing, we set $L_m$ as the maximum depth of the network which is determined by processing a BFS scheme. By using the same address space $A$ determined by our protocol, we can calculate a $C_m$ such that $\frac{C_mL_m+1}{C_m-1} < A$. Here, we make the value of $C_m$ at least as 2 even if the address space conducted by ZigBee will larger than ours too much. Fig. 12(a) shows the results. Although this method make ZigBee addressing have a better performance than that used in Fig. 11, ZigBee addressing still incurs many orphans. Moreover, as Fig. 12 shows, the value of $L_m$ will grow up to 80 when the transmission range is only 20 m. This means that the value of $C_m$ only can be 2 and the address space will be up to $2^{80}$. Therefore, if we want to let all nodes connect to the network and increase the $C_m$ of ZigBee, this will result in extremely larger address space and worse address utility.

VI. CONCLUSIONS

In this paper, we contribute in formally defining the PCC-WSN topology. Also, we have proposed a formation scheme to divide nodes into several paths and clusters. Then a two-level ZigBee-like hierarchical address assignment and routing schemes for PCC-WSN are conducted. The proposed address assignment scheme assigns each node both level-1 and level-2 addresses as its network address. With such a hierarchical structure, routing can be easily done based on addresses of nodes. We also show how to allow nodes to utilize shortcuts.
REFERENCES


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 got his Ph.D. in Computer and Information Science from the Ohio State University in January of 1994. He was/is Chairman (2005–2009), Chair Professor (2011–present), and Dean (2011–present), Department of Computer Science, National Chiao-Tung University, Taiwan. Dr. Tseng received Outstanding Research Award (National Science Council, 2001, 2003, and 2009), Best Paper Award (Int’l Conf. on Parallel Processing, 2003), Elite I. T. Award (2004), and Distinguished Alumnus Award (Ohio State University, 2005), and Y. Z. Hsu Scientific Paper Award (2009). His research interests include mobile computing, wireless communication, and parallel and distributed computing. An IEEE Fellow, Dr. Tseng serves/served on the editorial boards of IEEE Trans. on Vehicular Technology (2005-2009), IEEE Trans. on Mobile Computing (2006-2011), and IEEE Trans. on Parallel and Distributed Systems (2008-present).