Risk-Aware Distributed Beacon Scheduling for Tree-Based ZigBee Wireless Networks

Li-Hsing Yen, *Member*, *IEEE*, Yee Wei Law, and Marimuthu Palaniswami, *Senior Member*, *IEEE*

Abstract—In a tree-based ZigBee network, ZigBee routers (ZRs) must schedule their beacon transmission time to avoid beacon collisions. The beacon schedule determines packet delivery latency from the end devices to the ZigBee coordinator at the root of the tree. Traditionally, beacon schedules are chosen such that a ZR does not reuse the beacon slots already claimed by its neighbors, or the neighbors of its neighbors. We observe, however, that beacon slots can be reused judiciously, especially when the risk of beacon collision caused by such reuse is low. The advantage of such reuse is that packet delivery latency can be reduced. We formalize our observation by proposing a node-pair classification scheme. Based on this scheme, we can easily assess the risk of slot reuse by a node pair. If the risk is high, slot reuse is disallowed; otherwise, slot reuse is allowed. This forms the essence of our *ZigBee-compatible, distributed, risk-aware, probabilistic beacon scheduling algorithm*. Simulation results show that on average the proposed algorithm produces a latency only 24 percent of that with conventional method, at the cost of 12 percent reduction in the fraction of associated nodes.

Index Terms—IEEE 802.15.4/ZigBee, tree topology, beacon scheduling, convergecast.

1 Introduction

A Wireless Personal Area Network (PAN), as defined by IEEE 802.15.4 [1], comprises radio devices that are characterized by low power, low data rate, short communication range, and low cost. A PAN is initiated by a PAN coordinator. To facilitate periodic power-saving operations, PAN coordinators have the option to enable the beacon mode by defining a *superframe*. A superframe delineates the timing structure of activities in a PAN, and consists of an active and an inactive period (Fig. 1). The active period begins with a beacon frame, by which nearby devices can identify the presence of the coordinator and thereby join the PAN by making *association*[1] with the coordinator. Following the beacon is a number of time slots used for data exchange between the coordinator and associated clients. In the inactive period that follows the active period, no data traffic is expected in the PAN so devices can either enter power-saving mode or attempt communication with devices in other coexisting PANs.

IEEE 802.15.4 only supports star topologies for PANs. The ZigBee specification [2] further specifies how to organize a number of IEEE 802.15.4 devices into a tree or mesh network topology. In a ZigBee tree network, the root is called the ZigBee Coordinator (ZC), internal nodes are called ZigBee Routers (ZRs), and leaf nodes are referred to as ZigBee End Devices (ZEDs).

1. Association is a communication primitive on the MAC sublayer, specified by the standard, by which a device associates itself with a particular PAN [2]. We say "associate" and "join" interchangeably.

- *L.-H. Yen is with the Department of Computer Science and Information Engineering, National University of Kaohsiung, No. 700, Kaohsiung University Rd., Nan Tzu Dist., Kaohsiung 811, Taiwan, R.O.C. E-mail: lhyen@nuk.edu.tw.*
- *Y.W. Law and M. Palaniswami are with the Department of Electrical and Electronic Engineering, The University of Melbourne, Parkville, VIC 3010, Australia. E-mail: {ywlaw, palani}@unimelb.edu.au.*

Manuscript received 29 Sept. 2010; revised 27 Jan. 2011; accepted 18 Mar. 2011; published online 26 Apr. 2011.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2010-09-0449. Digital Object Identifier no. 10.1109/TMC.2011.88.

1.1 Problem Statement

In a ZigBee tree network, every ZC/ZR should periodically broadcast its own beacon. The ZC/ZR can then exchange data with its children in the active period associated with the beacon. Similarly, every ZR/ZED should track its parent's beacon to maintain time synchronization and exchange data with its parent. However, concurrent transmissions of beacon or data frames from multiple ZC/ZRs may cause collisions at intended receivers and give rise to reception failures—a phenomenon termed *beacon collision*.

We could shift active periods of neighboring ZC/ZRs to avoid potential beacon collisions. More specifically, when a new ZR joins the network, it determines the time offset of its beacon transmission relative to its parent's. *Beacon scheduling* refers to the process of determining the beacon transmission time of each ZR such that beacon collisions can be prevented. Fig. 1 shows an example of a beacon schedule.

A straightforward yet conservative beacon scheduling strategy is to preclude overlapping active periods between any pair of ZC/ZRs regardless whether an overlapping leads to beacon collisions or not [3], [4]. This approach is absolutely safe but may not be feasible when numerous ZRs are involved. Since it disallows concurrent transmissions of data packets by scattered ZRs, this approach also increases packet delivery latency from each device to the ZC, making it not suited for *convergecast* [5].

A more aggressive strategy is to allow overlapping active periods that are apparently collision free. Overlapping of active periods is disallowed when such arrangement *may* give rise to beacon collisions. For example, the specification [2] states that the active period of a ZR shall not overlap with that of any physical neighbor (i.e., any device within communication range) or of the parent of any physical neighbor. Prior work [4], [6], [7] even disallows the

1536-1233/12/$31.00 © 2012 IEEE Published by the IEEE CS, CASS, ComSoc, IES, & SPS

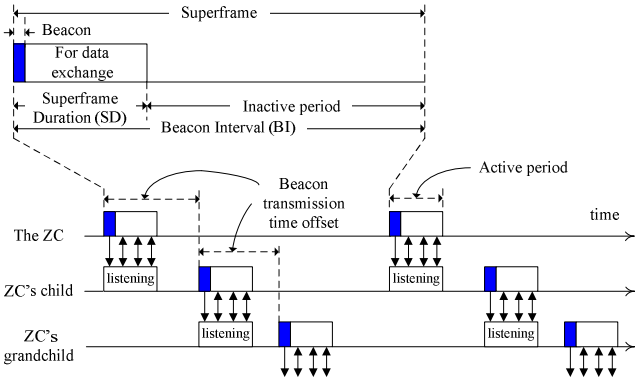


Fig. 1. Example of a beacon schedule and the structure of a superframe.

overlapping of active periods between two ZC/ZRs that have some physical neighbor(s) in common. For example, ZRs 4, 5, and 6 in Fig. 2 are not allowed to overlap their active periods with one another, since doing so causes beacon collision problem at ZR 7.

We make two observations regarding the absolute preclusion of beacon collisions caused by the overlapping of active periods:

1. *A beacon schedule that initially appears collision free may cause beacon collisions to devices joining the tree later.* Consider the example in Fig. 2. Suppose that the nodes are numbered according to the order they join the network. Then, ZR 6 is safe to overlap its active period with that of ZR 4 or 5 when it joins the tree. Assume that ZR 6 overlaps its active period with ZR 5's. When ZR 7 joins the network later, ZR 7 is unable to associate with either ZR 5 or 6 due to the beacon collisions caused by these two ZRs. A similar situation occurs when ZR 6 overlaps its active period with ZR 4's. This potential *risk* is inevitable since ZR 6 has no way of knowing future association events at the time of its association.
2. *The rule used to inhibit overlapping active periods may turn out to be too restrictive.* Using Fig. 2 as an example again, according to the rule, ZR 6 is not allowed to overlap its active period with ZR 2's, because these two nodes are physical neighbors. However, as long as neither ZR 6 nor ZR 2 has a child in the future, no node will suffer from beacon collisions between ZR 6 and ZR 2.

Our conclusion from the above observations is that the risk of beacon collisions caused by the overlapping of active periods can be completely eliminated only when full topology information of the tree is available. In practice, however, only partial topology information is available to ZRs at the time when they schedule their active periods. This limitation naturally gives rise to a nonzero risk of beacon collisions. The rule used by the specification or prior work to inhibit overlapping of active periods is also too restrictive, increasing packet delivery latency from each device to the ZC. A more practical strategy is to estimate the risk of beacon collisions and then according to the risk decide whether or not to allow overlapping active periods between a pair of nodes.

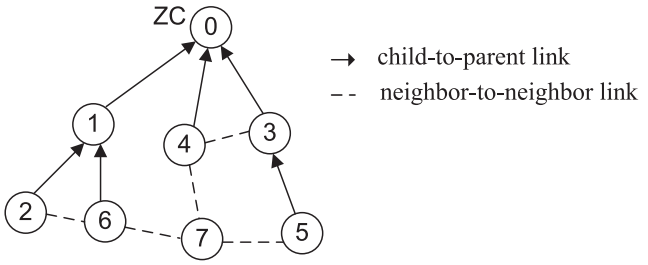


Fig. 2. A ZigBee network with a tree topology.

1.2 Contribution and Organization

Our contribution is a *ZigBee-compatible, distributed, risk-aware, probabilistic beacon scheduling algorithm* that allows a node to locally assess the risk of beacon collisions, and based on the assessed risk, to locally arrange its active period that minimizes the latency to its parent. We express such a risk as *risk probability*, that is, the probability that overlapping active periods between two nodes will cause beacon collisions as seen by a future joining node. To assess this risk probability, we classify node pairs into Inhibited Pairs (IPs), Visible Pairs (VPs), Hidden Pairs (HPs) and Uninhibited Pairs (UPs), and according to the pair type, calculate the corresponding risk probability. The novelty of our work lies predominantly in this classification scheme and the estimation of this risk probability.

The rest of this paper is organized as follows: Section 2 covers the background of the problem. Section 3 discusses related work. In Section 4, we describe our algorithm. Section 5 contains the simulation results that confirm the advantage of our algorithm. Finally, Section 6 concludes.

2 Preliminaries

IEEE 802.15.4 devices can be categorized into full function devices (FFDs) and reduced function devices (RFDs). FFDs are eligible to be PAN coordinators and are capable of forwarding frames for other PAN members. RFDs lack such capability and rely on FFD's frame-forwarding service to communicate with other devices in the PAN. In a ZigBee tree network, the ZC and the ZRs are functionally identical to PAN coordinators, and therefore can only be FFDs. RFDs, on the other hand, must join the tree as ZEDs.

In the beacon-enabled mode, the ZC and all ZRs broadcast beacon frames periodically. The time between two consecutive beacons is called Beacon Interval (BI), while the duration of an active period is the Superframe Duration (SD). BI and SD in IEEE 802.15.4 are defined as follows:

$$\begin{cases} \mathrm{BI} = aBaseSuperFrameDuration \times 2^{\mathrm{BO}} \\ \mathrm{SD} = aBaseSuperFrameDuration \times 2^{\mathrm{SO}} \end{cases},$$

where $aBaseSuperFrameDuration = 15.36$ ms for a data rate of 250 kbps in the 2.4 GHz frequency band, and Beacon Order (BO) and Superframe Order (SO) are two integers ranging from 0 to 14. Theoretically speaking, each ZR may have its own independent BI/SD setting [4]. Beacon scheduling in this context may extend over several superframes. For simplicity, we assume that all ZC/ZRs have identical BI and SD settings, and consider only schedules that fit into a single superframe. Furthermore, the whole BI

TABLE 1
Partial List of Symbols

Symbol	Semantics
n	Total number of nodes in the network
$2^{\text{BO-SO}}$	Number of beacon slots
Lm	Maximum depth of a tree
Cm	Maximum number of children of a ZC/ZR
Rm	Maximum number of children of a ZC/ZR that can be ZRs
$C(u)$	Set of u's children
r	Effective communications range
$d(u,v)$	Distance between nodes u and v
$N(u)$	$\{v \mid d(u,v) \leq r\}$
R	Region where ZigBee devices are deployed
A	Area of R
l	Width/height of R when R is square
p	Probability that a node is another node's physical neighbor
$J(u,v)$	Area of the region jointly covered by nodes u & v
$P(u,v)$	Expected probability of a future node being a victim of the slot reuse between u and v
$P_V(u,v)$	$P(u,v)$ for the case $(u,v) \in$ VP
$P_H(u,v)$	$P(u,v)$ for the case $(u,v) \in$ HP
$P_U(u,v)$	$P(u,v)$ for the case $(u,v) \in$ UP

can be divided into nonoverlapping time periods called *beacon slots*, each of which has a duration equal to SD. It follows that the number of available beacon slots in a BI is $2^{\text{BO}-\text{SO}}$. This value is typically large to yield an energy-conserving low duty cycle (between $\sim 0.1\%$ and $\sim 2\%$ regardless of the frequency band [2]).

The extent of a tree-based ZigBee network is controlled by parameters Lm, Cm, and Rm (Table 1). According to the ZigBee specification, the ZC is at depth 0 and devices at depth Lm can only be ZEDs, not ZRs. Let $T(\text{Lm}, \text{Rm})$ be the maximal possible ZigBee tree (disregarding ZEDs) that can be formed, given Lm and Rm. The number of ZC/ZRs in $T(\text{Lm}, \text{Rm})$ is $\sum_{i=0}^{\text{Lm}-1} \text{Rm}^i = \frac{\text{Rm}^{\text{Lm}}-1}{\text{Rm}-1}$. Therefore, if

$$2^{\text{BO-SO}} \geq \frac{\text{Rm}^{\text{Lm}} - 1}{\text{Rm} - 1}, \tag{1}$$

then each ZC/ZR can be assigned a unique beacon slot for nonoverlapping beacon transmissions. If (1) does not hold, which may happen in practice, some beacon slots must be reused to accommodate all possible ZRs in the network.

Slot reuse may give rise to beacon collisions. Beacon collisions may incur two different types of "harm," depending on whether the victim has already joined the network:

- **Transmission failure**. This happens when a device v interferes with other device w already in the tree by taking the same beacon slot of w's parent's upon v's arrival. Refer to Fig. 3 for a possible scenario. The victim in this case (device w) may suffer from the beacon collisions by experiencing high bit error rates, losses of the synchronization with its parent, or low link-layer throughput. This kind of harm is considered unacceptable and should be avoided by any beacon scheduling scheme.
- **Less associable ZC/ZRs**. The amount of ZC/ZRs accessible to a device joining the network may be lessened due to beacon collisions from two or more existing ZC/ZRs. Refer to Fig. 4 for a possible cause, where w is a common physical neighbor of u and v. We assume that all collided beacons and data frames are garbled. Therefore, w cannot associate with either u or v. Node w may still find and associate with another ZC/ZR, but there is a nonzero probability that w fails to join the tree due to the absence of other associable ZC/ZRs.

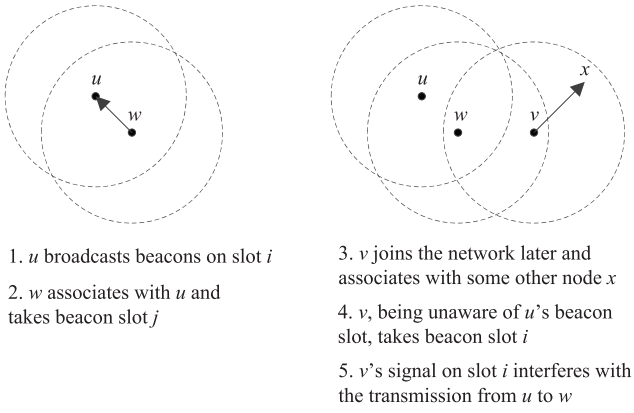


Fig. 3. An instance of transmission failure due to beacon collisions.

Regardless of the slot-reuse rule, the complete tree and physical topology must be available to make a slot-reuse schedule *absolutely* collision free. This requirement is considered impractical due to the way by which ZigBee trees are constructed. A device must hear beacon frames from a ZC/ZR before it can send a join request to the ZC/ZR. Before receiving the request, the ZC/ZR is not aware of the presence of that device. This means that ZC/ZR's beacon transmission schedule is determined with only partial knowledge of the network topology. Beacon collisions are thus inevitable. Fortunately, we can carefully reuse beacon slots to avoid the harm of transmission failure; the only price is lessened associable ZC/ZRs.

The way by which ZigBee trees are constructed also implies that any ZigBee-compatible beacon scheduling method should be an online algorithm by nature: *slot numbers are assigned while tree construction is in progress*. Our algorithm is ZigBee-compatible because it respects exactly this principle.

A ZED/ZR is allowed to send frames to its parent only during the parent's active period. So, a ZR in fact should participate in two active periods within a BI. The one

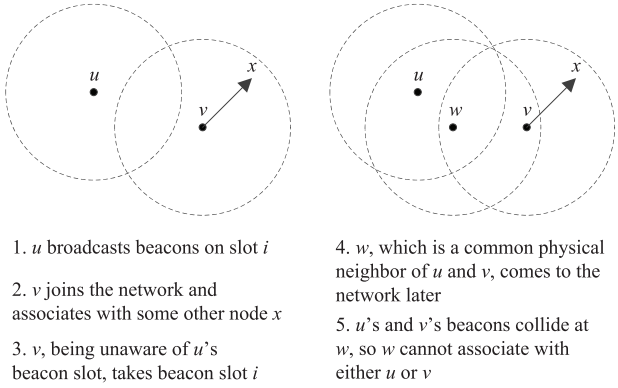


Fig. 4. An example of lessened associable ZC/ZRs due to beacon collisions.

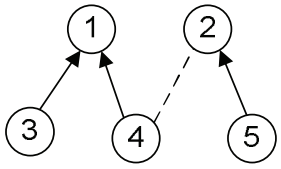


Fig. 5. An example illustrating the property of pairwise scheduling.

associated with its parent's beacon is to exchange data with its parent. The other, which is associated with its own beacon, is to exchange data with its child devices. If a ZR's active period is placed *after* its parent's in a schedule, all packets that it receives from its children during the current superframe will not be forwarded to its parent until the next superframe. Similarly, a ZR is unable to receive packets from its parent and then forward them to its children within the same superframe, if the ZR's active period is placed *before* its parent's. Therefore, uplink (from a device to its parent) and downlink (from a ZR to its child) packet delivery latencies are also determined by beacon scheduling. The total packet delivery latency along a path in the tree can be minimized if transmissions along the path are arranged in sequence. Such a schedule is called *cascading*.

For the ensuing discussion, most of the symbols used are summarized in Table 1.

3 Related Work

The beacon collision problem in ZigBee tree networks is similar to convergecast tree scheduling (CTS) problem in wireless sensor networks (WSNs). Convergecast is a communication primitive that enables the delivery and aggregation of data from each sensor node to a collecting point called sink. The CTS problem concerns how to utilize a tree rooted at the sink for convergecasting with Time Division Multiple Access (TDMA) used as the underlying medium access control (MAC) protocol. The objective is to allocate a time slot to each pair of nodes for data transmission so as to minimize the incurred latency while eliminating potential transmission collisions. This problem has been proven to be NP-hard [8]. Existing methods imposed different assumptions and requirements. Some approaches [5], [9] assume the use of Code Division Multiple Access (CDMA) combined with TDMA as the underlying MAC scheme so as to increase the possibility of simultaneous data transmissions. Some approaches generate a transmission schedule for every node after a tree is given or constructed [10], [11], [12], some determine the schedule before the tree construction [8], and some others generate the schedule while the tree is under construction [5], [9].

Despite these design variations, two common properties of the aforementioned approaches make them not ZigBee-compatible. The first is the adoption of *pairwise scheduling* model, which demands one time slot for *each* pair of nodes. In contrast, each ZR in a ZigBee tree exchanges data with *all* of its child nodes within the same active period. Therefore, only one schedule can be arranged for the one-to-many communication between a ZR and all its children. Fig. 5 illustrates the difference between pairwise scheduling and the scheduling for ZigBee tree networks. In pairwise scheduling, node 3 there can reuse the time slot of node 5 to transmit its data to node 1 as there is no conflict between them. However, in a ZigBee tree network, transmissions of data from ZR 3 and ZR 4 to ZR 1 take place at the same active period (the beacon slot of ZR 1). Therefore, if we overlap the data transmission from ZR 3 to ZR 1 with that from ZR 5 to ZR 2, the transmission from ZR 4 to ZR 1 may collide with that from ZR 5 to ZR 2.

The second property is the requirement of full knowledge of *conflict* or interference relationship among nodes to yield an interference-free schedule. A commonly adopted model is to assume that two nodes that are separated by n hops or less in the physical topology or with a distance of d unit length or less will interfere with each other. This requirement is not practical in our context as only partial knowledge of conflict or interference relationship is available to ZRs when they schedule their beacon slots.

The beacon collision problem in ZigBee tree networks has not been addressed until recently. To avoid potential beacon collisions, the Beacon-Only Period approach [4] lets all ZC/ZRs synchronize to the same superframe structure with variety in beacon transmission time. More specifically, an additional time window for contention-free beacon transmissions is added to the beginning of each superframe to prevent possible beacon collisions. Beacon-Only Period demands a considerable change to the standard and is therefore not ZigBee-compatible.

An intuitive approach to ZigBee-compatible beacon scheduling is to completely avoid reusing beacon slots. In [3], the ZC allocates an exclusive beacon slot to each ZR upon the ZR's associations with the network. Since all ZRs use exclusive beacon slots, beacon collisions are impossible but the number of ZRs that can be accommodated by the network is limited. The resultant schedule is also not optimal in terms of message latency. Superframe Duration Scheduling (SDS) [4] assumes a configuration with heterogeneous BI/SD settings. It determines whether the given configuration is schedulable, and provides a collision-free schedule that may span several superframes when the answer is positive. However, SDS still does not consider reusing beacon slots.

Spatial reuse of beacon slots is the key to minimizing the latency of data delivery. Koubâa et al. [4] suggested using the ZC to collect location information of all ZRs and determine which group of ZRs can use the same beacon slot without beacon collisions. They presented the rule for slot reuse but did not detail how to realize it. Pan and Tseng [6] investigated the problem of finding a collision-free schedule for ZigBee tree networks that minimizes the maximal convergecasting latency. They proved that this problem is NP-hard and proposed two heuristic scheduling algorithms, namely centralized tree-based assignment and distributed slot assignment (DSA). Centralized tree-based assignment requires complete topology information as input and is not much better than DSA. In DSA, each node u chooses the slot that gives the lowest latency with respect to u's parent, but at the same time does not collide with the slots occupied by the nodes in u's $2r$-neighborhood (r is the radio range).

In [7], the authors studied how to minimize the latency of broadcasts from the ZC to every end device as well as convergecast latency at the same time. The resultant

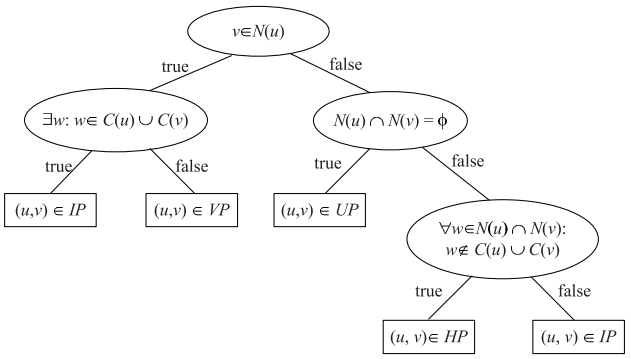


Fig. 6. Proposed classification tree for identifying the type of a ZC/ZR pair (u, v).

schedule demands a modification on the original ZigBee superframe structure to allow each ZR to participate in four active periods within a BI, two for cascading convergecast schedules and the other for cascading broadcast ones. This work also considers the same $2r$-neighborhood interference model.

We argue that the rule imposed by prior work for slot reuse is, however, too restrictive. The point is that reusing a beacon slot should be safe as long as there is no victim of such reuse. The schemes in [4], [6], [7] avoid allocating a ZR's beacon slot to another ZR if they are physical neighbors or have physical neighbors in common. By this rule, ZRs 1 and 3 in Fig. 2 are not allowed to take the same beacon slot, since they share a common physical neighbor (i.e., the ZC). However, any ZR's beacon slot is for this ZR to exchange data with its children, not for data exchange with its parent. Since child nodes of ZRs 1 and 3 do not interfere with each other, reusing ZR 1's beacon slot at ZR 3 in fact does not make any victim. This kind of slot reuse is permitted by the ZigBee specification. However, the specification inhibits slot reuse between physical neighbors, though such a reuse does not necessarily make a victim. The ZR pair $(2, 6)$ in Fig. 2 is an example. As long as neither ZR 2 nor ZR 6 has a child in the future, no node will suffer from the slot reuse between ZR 2 and ZR 6.

In a preliminary version of this paper [13], we have presented a systematic approach for identifying the cases when slot reuse is relatively safe. This is done by classifying node pairs into different *pair types*, and calculating the associated *risk probability*. High-risk probability means the corresponding node pair would at a high probability prevent a future neighboring node from joining, and the node pair should therefore not share a beacon slot. Conversely, low-risk probability means it is relatively safe for the node pair to share the same beacon slot. This paper enhances our previous work by developing a more rigorous analysis on risk probabilities, detailing the algorithm for parent and slot selections with implementation details, and performing more thorough simulations.

4 Probabilistic Risk-Aware Beacon Scheduling

In this section, we first describe our node-pair classification scheme, then our risk probability assessment methodology, and finally our ZigBee-compatible, distributed, risk-aware, probabilistic beacon scheduling algorithm. The node-pair classification scheme and risk probability assessment methodology allow the algorithm to reduce latency by reusing beacon slots, while still minimizing the risk of new nodes not being able to join the tree.

TABLE 2
Comparison of Slot-Reuse Policies

Beacon slot reusable?	IP	VP	HP	UP
Prior work [4], [6], [7]	No	No	No	Yes
The specification [2]	No	No	Yes	Yes
Our rule	No	Yes	Yes	Yes

4.1 Node-Pair Classification

Our classification rule of node pairs demands (and only demands) local knowledge of neighborhood and parenthood relations. We use $N(u)$ to denote the set of u's physical neighbors. Assuming symmetric communication model, we have $v \in N(u) \Longleftrightarrow u \in N(v)$ for all device pair (u, v). We also denote by $C(u)$ the set of u's children in the ZigBee tree that u joins. Note that $C(u) \subset N(u)$. We classify a ZC/ZR pair (u, v) as one of the following types (Fig. 6):

- **Inhibited pair**. u and v are physical neighbors, and either u or v has a child; or, u and v are not physical neighbors, but u and v have a common neighbor which is a child of either u or v.
- **Visible pair**. u and v are physical neighbors but neither u nor v has any child.
- **Hidden pair**. u and v are not physical neighbors but have physical neighbors in common, although all these physical neighbors are neither u's nor v's children.
- **Unrelated pair**. u and v are not physical neighbors, neither do they have physical neighbors in common.

The difference among various slot-reuse policies, including ours, is summarized in Table 2. From this table, we can see our policy reuses beacon slots to the furthest extent.

If an instance of slot reuse causes beacon collisions to devices (ZRs or ZEDs) already in the network, we call such reuse *damaging* in the sense that it may cause transmission failures. A slot reuse between a pair of nodes (u, v) is damaging only if u's and v's beacons can collide at a device w that is already a child node of either u or v. Therefore, slot reuses by IP are damaging (we have already seen an example in Fig. 3) and thereby not allowed by any known beacon scheduling scheme.

A slot reuse between a pair of nodes (u, v) is *risky* if it can lessen accessible ZC/ZRs for a future node w. One possible cause for the lessening is that w happens to be a common physical neighbor of u and v (Fig. 4). For this cause, slot reuses by IP, VP, and HP are all risky. A slot reuse by $(u, v) \in$ UP is risky only if u and v are close enough to allow a common physical neighbor in between. However, the absence of such common physical neighbor for now (which qualifies u and v as an UP) does not imply the existence or nonexistence of such neighbor in the future. This is why the corresponding entry in Table 3 is marked "Maybe."

For slot reuses by $(u, v) \in$ VP, there is another cause for the lessening of accessible ZC/ZRs. A future node (say, w) may hear beacon signal from one of these two nodes (say, u)

TABLE 3
Harm of Slot Reuses by Different Types of Node Pairs

	Damaging?	Risky?
IP	Yes	Yes
VP	No	Yes
HP	No	Yes
UP	No	Maybe

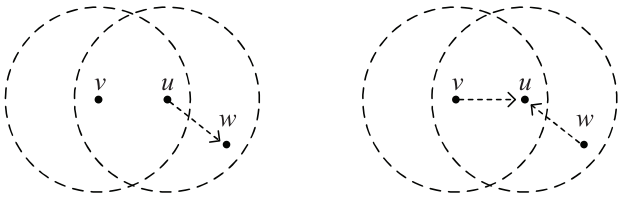
but not from the other (v is a hidden terminal to w). Because beacon signal is broadcasted unidirectionally, the beacon signal from u to w is not interfered by v's transmission. However, if w transmits association requests to u, the requests can collide with v's transmission at u on the same beacon slot. Refer to Fig. 7 for an illustration. The collision causes the failure of the association between w and u and thus lessens the amount of ZC/ZRs that are associable to w.

In summary, as opposed to existing proposals, we propose reusing beacon slots by VPs, HPs, and UPs when the risk is low. Given any *online* beacon scheduling algorithm that uses only local topological information, there is no guarantee that a good schedule would not become a bad schedule when a new node joins the network later, but our approach is to minimize such probability.

4.2 Assessment of Slot-Reuse Risk

A key task in our algorithm is to estimate the risk probability of slot reuse for a given device pair (u, v) based on the current neighborhood and parenthood knowledge. Our estimate does not assume the knowledge of the physical distance between u and v, or the total number of nodes. We assume that the deployment of ZigBee devices is random yet follows a uniform distribution over a region R. Let A be the size (area) of R. Each device is assumed to have an *effective* communications range of r. A piece of area in R is said to be covered by a device u and termed u's coverage if every point in this area is within the effective communications range of u. We use $J(u, v)$ to denote the size of the region jointly covered by two devices u and v, and use $d(u, v)$ to denote the physical distance between them.

It is well known that the radio coverage of a wireless transmitter is not a perfect disk. A more realistic signal propagation model may consider random variations in path loss at different locations [14], [15] or in different directions [16]. Consequently, data transmission between a pair of nodes becomes a probabilistic event with probability distribution being a function of the distance or direction between the transmitter and the receiver. The proposed node-pair classification and associated analyses are based on the abstraction of physical neighbors, which is a *binary* relation in the sense that two nodes are either physical neighbors or are not. Such a relation requires the setting of a threshold on packet reception probability for the judgment of physical neighbors with respect to a transmitter. Consequently, the transmitter's effective communications range would be irregular. Nevertheless, we can always find a lower bound of the transmission range, within which any device can certainly communicate with the transmitter, or an upper of the range, beyond which no devices are able to successfully receive the transmitter's data. The notation r in our definition can be interpreted as the lower bound, the upper bound, or the mean value of the transmission range, in which case the derived analytic results (presented in the following) become essentially the best base, the worst case, or the average-case risk values.



w can hear u's beacon signal but not v's

w's transmission collides with v's at u

Fig. 7. Potential data collisions due to the slot reuse between $(u, v) \in$ VP.

Lemma 1. *The expected area of the region jointly covered by two uniformly distributed nodes u and v is*

$$\mathrm{E}[J(u,v)] = \begin{cases} \left(\pi - \dfrac{3\sqrt{3}}{4}\right) r^2 & \text{if } d(u,v) \le r, \\ \dfrac{\sqrt{3}}{4} r^2 & \text{if } r < d(u,v) \le 2r. \end{cases}$$

The proofs of Lemma 1 as well as other theorems are given in the Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TMC.2011.88.

A node placed near the boundary of R will cover less area than expected, as part of its coverage is outside R. This is referred to as the border effect. To avoid clumsy results caused by the border effect, the following analysis assumes that the region covered by any node is completely within R. If R is a rectangle area, the assumption can be achieved by adopting the torus convention [17], which turns a flat rectangle into a torus. With this assumption, the probability of link occurrence [18] is $p = \pi r^2 / A$. Our core result consists of Theorems 1-3.

Theorem 1. *Assume nodes are uniformly distributed in a region R of size A, and $A \gg \pi r^2$. Suppose that a pair of nodes $(u, v) \in$ VP are using the same beacon slot when a new node w joins. Denote by $P_V(u, v)$ the expected probability that w suffers from the harm caused by the slot reuse between u and v. We have $P_V(u, v) = (1 + \frac{3\sqrt{3}}{4\pi})p$, where $p = \pi r^2 / A$.*

Theorem 1 is proven with the help of the inclusion-exclusion principle and the result of Lemma 1. It indicates that $P_V(u, v) \approx 1.41p$, so a large communications range implies a high probability of beacon collisions between nodes in a VP.

Theorem 2. *Denote by $P_H(u, v)$ $(resp.\ P_U(u, v))$ the expected probability that a joining node w becomes a victim of slot reuse between u and v when $(u, v) \in$ HP (resp. $(u, v) \in$ UP). Furthermore, denote by $P^*_U(u, v)$ the expected probability that a joining node w becomes a victim of slot reuse between u and v when $(u, v) \in$ UP, with the additional condition that $r < d(u, v) \le 2r$. Let k be the number of neighbors of v. Then, $P_H(u, v)$ and $P_U(u, v)$ are related to $P^*_U(u, v)$ by*

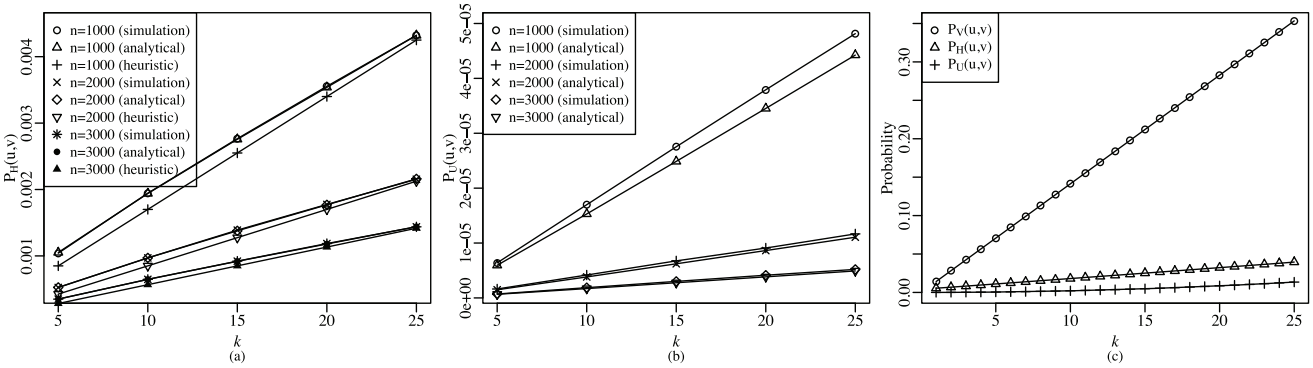


Fig. 8. (a) Comparison of simulation, analytical and heuristic values of $P_H(u,v)$; note the simulation and analytical values virtually overlap. (b) Comparison of simulation and analytical values of $P_U(u,v)$. (c) Comparison of $P_V(u,v)$, $P_H(u,v)$, and $P_U(u,v)$ for $n = 100$. The simulation methodology is explained in the text.

$$P_H(u,v) = \frac{\sqrt{3}}{4\pi(1-\varphi(k))}p - \frac{\varphi(k)}{1-\varphi(k)}P_U^*(u,v), \qquad (2)$$
$$P_U(u,v) = 3pP_U^*(u,v),$$

where $p = \pi r^2/A$ *and*

$$\varphi(k) = \frac{2}{3}\int_{\theta=0}^{2\pi/3}\left[1-\frac{\theta-\sin\theta}{\pi}\right]^k \sin\theta\, d\theta.$$

Function $\varphi(k)$ is the expected probability of all v's k neighbors not falling in the region jointly covered by u and v with the condition that $r < d(u,v) \le 2r$. Since $\varphi(k)$ is decreasing with the value of k (see Fig. 13 in the Appendix, available in the online supplemental material), $\varphi(k) \le \varphi(1) = 1 - \frac{\sqrt{3}}{4\pi}$ and $1-\varphi(k) \ge 1-\varphi(1) = \frac{\sqrt{3}}{4\pi}$ for all $k \ge 1$. It follows that the first term of (2) is less than $1p$ and $P_H(u,v)$ is therefore upper bounded by p. Moreover, for the case of $k = 1$,

$$P_H(u,v) = p - \frac{1-\frac{\sqrt{3}}{4\pi}}{\frac{\sqrt{3}}{4\pi}}P_U^*(u,v) \ge 0.$$

It follows that

$$P_U^*(u,v) \le \frac{\sqrt{3}}{4\pi-\sqrt{3}}p \approx 0.16p.$$

Therefore, $P_U(u,v) = 3pP_U^*(u,v) \le 0.48p^2$.

For a general k, the integration operation of $\varphi(k)$ is computationally intensive. Fortunately, values of $\varphi(k)$ corresponding to practical values of k could be precomputed and stored in the nodes before deployment. These values can then be looked up whenever $\varphi(k)$ is to be calculated. More importantly, according to Theorem 2, if we can find $P_U^*(u,v)$, we can find exact values of $P_H(u,v)$ and $P_U(u,v)$. Theorem 3 allows us to compute $P_U^*(u,v)$.

Theorem 3. *Using the definitions of Theorem 2,*

$$P_U^*(u,v) = \int_r^{2r} J(x)\left\{\frac{\left[1-\frac{J(x)}{\pi r^2}\right]^k \frac{2x}{3r^2}}{\int_r^{2r}\left[1-\frac{J(x)}{\pi r^2}\right]^k \frac{2x}{3r^2}}\right\}dx,$$

where

$$J(x) = r^2\left[2\arccos\frac{x}{2r} - 2\sqrt{1-\left(\frac{x}{2r}\right)^2}\left(\frac{x}{2r}\right)\right].$$

Fig. 8a (resp. Fig. 8b) compares the values of $P_H(u,v)$ (resp. $P_U(u,v)$) obtained using Theorems 2 and 3 with the values obtained from simulations. In the simulations, the simulation area is fixed, the effective communications range is set as $\sqrt{kA/(n\pi)}$, and nodes are added to the simulation area until an average node degree of k is reached (consequently, the total number of nodes is *approximately* n). While the analytical results give a high degree of accuracy, Theorem 3 becomes more computationally demanding as k increases. For sensor nodes to estimate $P_U^*(u,v)$ efficiently, we propose the approximation

$$P_H(u,v) \approx 0.17p. \qquad (3)$$

The determination of the constant in (3) is given in the Appendix, available in the online supplemental material. The values of $P_U(u,v)$ are generally so low that they are considered to be zeros in the algorithm described in Section 4.3. In general, $P_V(u,v) \ge P_H(u,v) \ge P_U(u,v)$ (see Fig. 8c).

4.3 Parent and Slot Selections

We first describe our core algorithm, and then variations of the core algorithm. The ZC by default claims beacon slot 0. When a node u wants to join the network, it should first scan for beacons from neighboring ZC/ZRs that are already in the tree. These ZC/ZRs comprise the set of all parent candidates denoted by Φ. From Φ, all ZC/ZRs that have no room to accommodate any child[2] are first removed. Device u then sorts Φ according to the parent-selection policy. According to the ZigBee standard [2, p352], u's parent should be the neighboring ZC/ZR that has the minimal depth value. Here, we propose to select the neighboring ZC/ZR that has the maximal slot number, and use depth value to break ties. We shall explore other parent-selection policies via simulations in the next section. After Φ has been sorted, u selects one member from Φ as its parent by running the following procedure:

1. Starting from the first-ranking member ϕ of Φ, check if u can associate with ϕ as a ZR. If ϕ has less than

2. A device $\phi \in \Phi$ has no room to accommodate any child if ϕ already has Cm children or has depth Lm.

Rm ZRs among its children, and has depth $(\mathrm{Lm}-2)$ or less, and pickSlot(ϕ, u) returns a nonnegative slot number s, then u associates with ϕ as a ZR and starts using beacon slot s. Otherwise, repeat the check with the next-ranking member of Φ until all members of Φ have been checked. Note that the depth check in this step is for determining whether u can join as a ZR or only as a ZED.

2. If there is no member of Φ that u can associate with as a ZR, then u associates with the first-ranking member of Φ as a ZED.

The pseudocode of the function pickSlot() in the procedure above is given in Algorithm 1. On line 4 of the algorithm, s is iterated starting from the closest value to s_ϕ until a suitable value is found. This design is for efficient convergecast.

Algorithm 1. pickSlot(ϕ, u)

```
 1: Let s_φ be φ's slot
 2: Let s_max be total number of slots
 3: for i = 1 to total s_max − 1 do
 4:     s ← (s_φ − i) mod s_max
 5:     harm ← 0
 6:     for v ∈ N(u) or v ∈ {N(w)|w ∈ N(u)} do
 7:         if v's slot equals s then
 8:             Evaluate the risk of re-using v's slot
 9:             pair type of (u, v) ← getPairType(u, v)
10:             if (u, v) ∈ IP then
11:                 harm ← 1
12:             else if(u, v) ∈ VP then
13:                 harm ← 1 at probability P_V
14:             else if (u, v) ∈ HP then
15:                 harm ← 1 at probability P_H
16:             else if (u, v) ∈ UP then
17:                 harm ← 1 at probability P_U
18:             end if
19:         end if
20:         if harm = 1 then
21:             break   ▷ Continue evaluating remaining slots
22:         end if
23:     end For
24:     if harm = 0 then
25:         Neighbors' slots are risk-wise compatible
            with s, so
26:         return s
27:     end if
28: end for
29: Return -1          ▷ Negative value signifies failure
```

The proposed algorithms assume that u has the following information about its neighbors:

1. their IDs;
2. their slot numbers;
3. whether they have children;
4. their parent's ID;
5. their neighbors' IDs;
6. their neighbors' slot numbers.

With this information, getPairType() can detect all IP, VP, and HP but not UP (see Algorithm 2). Note that if u and v are an UP, then v can neither be an immediate neighbor nor a second-hop neighbor of u, and the function getPairType() would not even be called.

TABLE 4
Algorithmic Variants

Slot reuses by	Distributed			Centralized			
VP	✓	×	×	✓	×	×	×
HP	✓	✓	×	✓	✓	×	×
UP	•	•	•	✓	✓	✓	×
Name	DVHU	DHU	DU	CVHU	CHU	CU	C

Legend:
- ✓ accepts slot reuse probabilistically
- × rejects slot reuse
- • accepts implicitly

Algorithm 2. getPairType(u, v)

```
 1: if v ∈ N(u) then                          ▷ u, v are neighbors
 2:     if u has children or v has children then
 3:         return IP                  ▷ Left-most leaf in Fig. 6
 4:     else
 5:         return VP
 6:     end if
 7: else                                      ▷ u, v are not neighbors
 8:     hasCommonNeighbor ← 0
 9:     for each w ∈ N(u) do
10:         if v ∈ N(w) then
11:             if w ∈ C(u) or w ∈ C(v) then
12:                 return IP          ▷ Right-most leaf in Fig. 6
13:             end if
14:             hasCommonNeighbor ← 1
15:         end if
16:     end for
17:     if hasCommonNeighbor = 1 then
18:         return HP
19:     else
20:         return UP
21:     end if
22: end if
```

Table 4 lists all the variants of the aforementioned core algorithm. The variants are labeled according to whether they are distributed (D) or centralized (C), and which pair type they allow for slot reuse (V, H, U). For example, the distributed algorithm that allows slot reuse by VPs (probabilistically), HPs (implicitly), and UPs (implicitly) is labeled DVHU. Note that all variants reject slot reuse by IPs. To reject slot reuse by VPs/HPs/UPs, the probability P_V/P_H/P_U on line 13/15/17 of Algorithm 1 has to be set to 1. We elaborate on the variants as follows:

- **DVHU, DHU, and DU**. Distributed algorithms cannot detect UPs; hence, these variants accept slot reuse by UPs implicitly. DU rejects both VPs and HPs, and in this sense it implements the slot reuse strategy proposed by prior work [4], [6] in the literature. Like DSA [6], these variants make a node aware of the slots occupied by the node's two-hop neighbors, but unlike DSA, these variants do not achieve this by doubling the radio range from time to time. Doubling the radio range may confuse some nodes about who their neighbors really are;

TABLE 5
Beacon Fragment Format for Distributed Algorithms

Number of octets: 1	multiple of 9
Beacon fragment ID	(Neighbor ID, slot number) pairs

Beacon fragment ID:

Bits: 0	1–3	4	5–7
1 if this is the last beacon fragment; 0 if otherwise	Order of the beacon fragment. Supports up to 8 beacon fragments	1 if sender has children; 0 if otherwise. Defined for the first beacon fragment only	Reserved

(Neighbor ID, slot number):

Number of octets: 8	1
Neighbor's 64-bit IEEE extended address, which is also the parent's address if this is the first address in the first beacon fragment	Neighbor's slot number

furthermore, some RF transceivers such as the CC2480 may not support adjustment of transmission power. Instead, in DVHU, a node embeds information about its neighbors' slots in the beacon payload in the manner described in Section 4.4.

- **CVHU, CHU, CU, C**. With complete topology information, a ZC can detect all pair types. From a practical viewpoint, these variants are much less scalable and much less energy-efficient than the distributed variants. We only mention them here for comparison with the distributed variants later.

In summary, our core algorithm is ZigBee-compatible because it can be executed while the ZigBee tree itself is being formed. It is distributed because only local information is required. It is risk aware, because it can calculate the risk of slot reuse by using Theorems 1 and 2. It is probabilistic because it accepts/rejects slot reuse at a probability exactly dictated by the calculated risk.

4.4 Implementation Detail

The proposed algorithms require a node to broadcast information about its neighbors (see Section 4.3) in its beacon frames. The beacon frame field that can be used for its purpose is the *beacon payload*. In IEEE 802.15.4-2003, the beacon payload is an optional field used for security suite specification [1, p120]. In ZigBee-2007 (which is based on IEEE 802.15.4-2003), the beacon payload is used for signaling the network (NWK) layer instead [2, p. 414]. In both cases, the beacon payload is processed on the NWK layer, and can be up to 52 bytes long [1, p. 134]. For our purpose, we still use the first 15 bytes of the beacon payload for signaling the NWK layer, but the remaining 37 bytes to carry information required by the proposed algorithms. This arrangement requires the NWK layer to be modified, but on the protocol level, this modification remains compatible with ZigBee-2007.

Thirty seven bytes in the beacon payload is however not enough to carry all relevant information about a node's neighbors. As a remedy, we propose distributing the information in a number of "beacon fragments." One beacon fragment refers to the beacon payload of a single beacon frame. One beacon fragment is broadcast in a beacon slot. Table 5 shows the beacon fragment format that supports up to eight beacon fragments, or equivalently 32 neighbors (since each beacon fragment can record information about four neighbors at most). The proposed format also supports at most $2^{\text{BO-SO}} = 2^8$ beacon slots (since 1 byte is allocated to the slot number). The limit on the number of beacon slots is reasonable since the standard states: [2, p. 407]

> "It is recommended that a tree network utilize a superframe order of 0, ... and a beacon order of between 6 and 10, ..."

5 Simulation Results

The purpose of the simulations is

1. to compare the algorithms listed in Table 4;
2. to compare the policies for choosing a parent;
3. to compare DVHU with DSA [6].

Metrics. For measuring latency, two metrics are defined: *average maximum latency* [6] and *average latency*. Average maximum latency is defined as $\sum_i L_i/\max(i)$, where i indexes simulation runs, and L_i is the maximum latency corresponding to the ith simulation run. Average latency is the more conventional notion of the average of average latencies of all simulation runs. We are generally interested in 1) how these metrics vary with network size when network density is fixed; and 2) how these metrics vary with network density when the simulation area is fixed. To be explicit, we define network density as the average number of neighbors a node has.

Simulation mechanics. Nodes are pseudorandomly distributed in a square simulation area of $l \times l$ (l is the length of the area) with the ZC in the center. When the network density is fixed, nodes are added to the simulation area until the average number of neighbors per node reaches the specified level. The torus convention is used to cope with border effects.

5.1 Comparison of the Algorithms in Table 4

As evident in Fig. 9, there are four performance categories:

1. Algorithms DVHU and CVHU perform similarly (for reasons explained later) and are the best performers.
2. Algorithms DHU and CHU perform similarly and are quite worse than the previous category. This observation confirms that allowing slot reuse by VPs judiciously, as the previous category does, reduces latency.
3. Algorithms DU and CU perform similarly and are much worse than the previous category. This observation confirms that allowing slot reuse by HPs judiciously, as the previous category does, reduces latency.
4. Algorithm C is the worst performer since it does not allow any slot reuse at all. This serves as the benchmark for the worst performers in beacon scheduling algorithms.

Fig. 9 shows that the centralized algorithms offer hardly any advantage over the distributed algorithms. This outcome can be explained as follows: Recall that since distributed algorithms cannot detect UPs, they accept slot

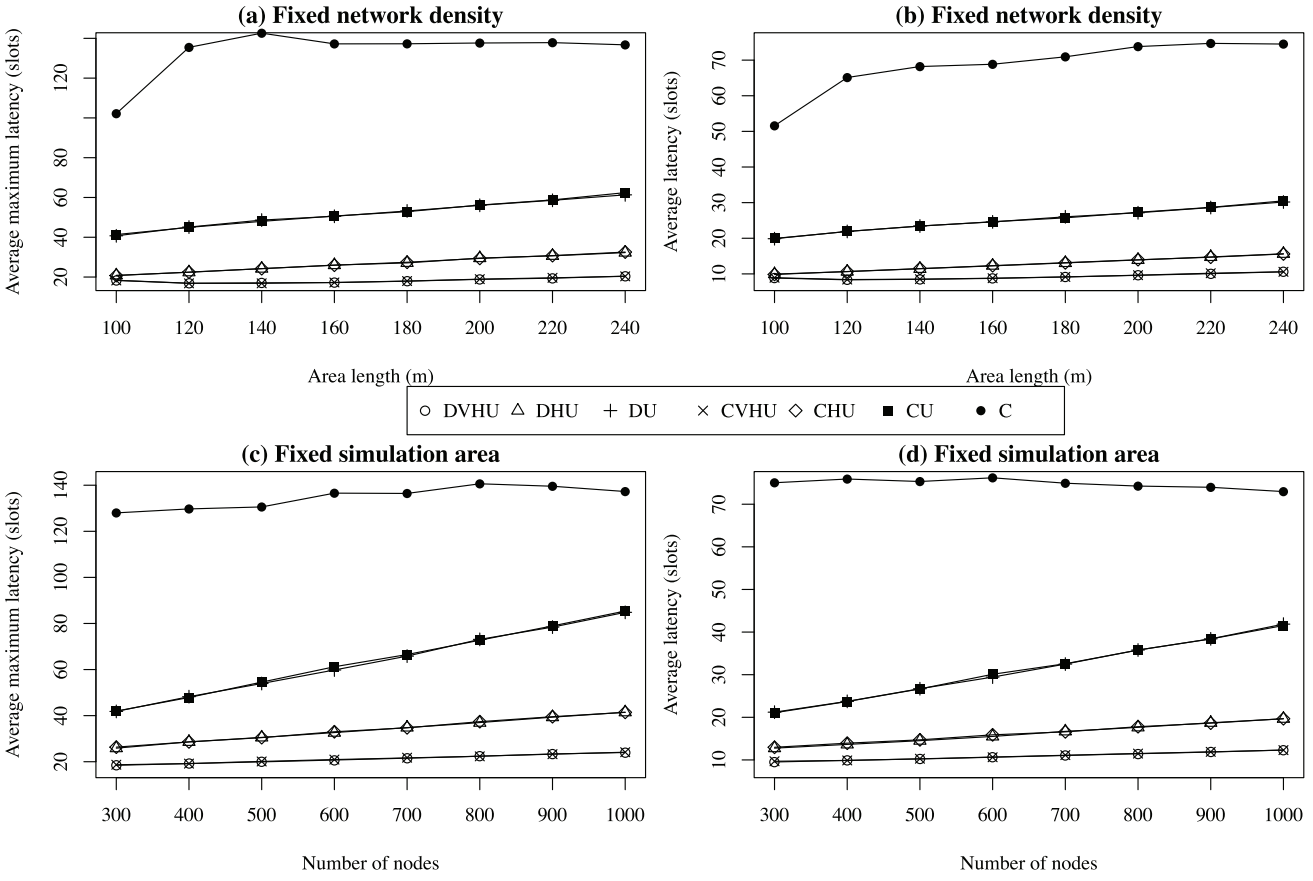


Fig. 9. Comparison of the algorithms in Table 4 in terms of average maximum latency and average latency. For fixed network density, $r = 25$ m, $k = 20$. For fixed simulation area, $l = 200$ m, $r = 20$ m. For all simulations, $\mathrm{Lm} = \infty$, $\mathrm{Cm} = 7$, $\mathrm{Rm} = 7$, $2^{\mathrm{BO-SO}} = 128$.

reuse by UPs implicitly. Meanwhile, the centralized algorithms can detect UPs, but the risk probability of slot reuse by UPs (P_U) is typically so close to zero (see Fig. 8 for examples) that the centralized algorithms almost always accept slot reuse by UPs. With identical policy toward VPs and HPs, and almost identical policy toward UPs, a distributed algorithm and its centralized counterpart are expected to perform almost identically in the latency metrics.

Commenting on the metrics themselves, we note that by virtue of Fig. 9, both metrics average maximum latency and average latency yield similar comparison of the algorithms. However, average latency gives lower standard deviations (data not shown). In the following, we will continue using average latency and drop average maximum latency.

5.2 Comparison of Parent-Selection Policies

In the previous simulations, among the neighbors which satisfy the Rm/Lm/Cm constraints, a node chooses the neighbor with the minimum tree depth as the parent, consistent with the ZigBee standard [2, p352]. However, this policy does not always lead to a tree configuration with the least average latency. We explore three other parent-selection policies. The four parent-selection policies being compared are hence:

- **Depth**. The neighbor with the minimum tree depth is chosen as the parent.
- **Depth then slot**. Neighbors are sorted primarily in ascending order of the tree depth, and secondarily in descending order of the slot number. The first-ranking neighbor is chosen as the parent.
- **Slot**. The neighbor with the maximum slot number is chosen as the parent.
- **Slot then depth**. Neighbors are sorted primarily in descending order of the slot number, and secondarily in ascending order of the tree depth. The first-ranking neighbor is chosen as the parent.

Fig. 10 shows that the slot-then-depth policy results in the least average latency, while using the depth parameter alone as suggested by the standard actually results in the highest average latency. While the slot-then-depth parent-selection policy recommended here is not the same as the depth-only policy specified in the standard, any ZigBee implementation that implements our policy remains interoperable with other ZigBee implementations, and thus should be considered standard compliant.

An interesting feature of Fig. 10a is the bend between $l = 100$ m and $l = 140$ m (which also exists but not as obvious in Figs. 9a and 9b). The reason can be understood as follows: To maintain a constant network density with a fixed radio range, the number of nodes $n \propto l^2$. As l increases from 100 to 120 m, n increases by 44 percent. At $r = 25$ m, many of the 44 percent additional nodes are still distributed close to the ZC in the center. This results in lower average latency. As l increases further, the nodes become more dispersed from the center and the average latency starts to increase.

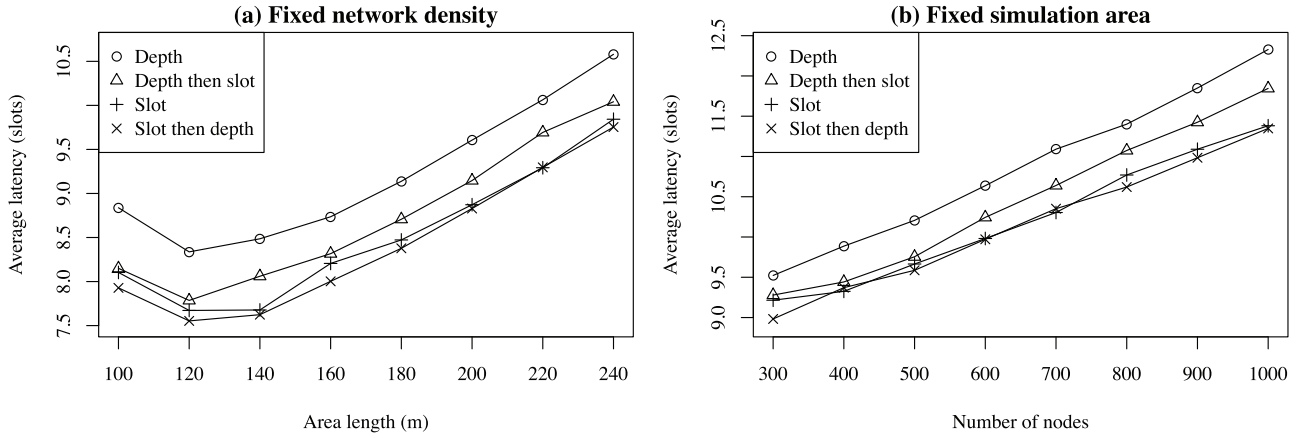


Fig. 10. Comparison of parent-selection policies using DVHU. For fixed network density, $r = 25$ m, $k = 20$. For fixed simulation area, $l = 200$ m, $r = 20$ m. For all simulations, $\mathrm{Lm} = \infty$, $\mathrm{Cm} = 7$, $\mathrm{Rm} = 7$, $2^{\mathrm{BO\text{-}SO}} = 128$.

5.3 Comparison of DVHU with DSA

Here, we compare DVHU with DSA in terms of average latency and fraction of associated nodes (i.e., the ratio of the number of associated nodes to the total number of nodes). Changing Lm clearly affects the average latency. Changing Lm also affects fraction of associated nodes, as a "taller" tree admits more devices into the network. Fig. 11 shows that DVHU outperforms DSA for $\mathrm{Lm} \in \{3, 6, 9\}$ in terms of average latency. On average, the average latency with DVHU is only 24 percent of that with DSA. In terms of fraction of associated nodes, DVHU can admit less devices than DSA, due to its aggressive slot-reuse policy. For example, when a VP uses the same beacon slot, the pair can no longer admit any child. On average, the fraction of associated nodes with DVHU is 12 percent lower than that with DSA.

In Section 2, we mentioned that a node might be "unaccommodated" when two or more of its neighbors use the same beacon slot. To quantify and thereby visualize this possibility of unaccommodation, we define the ratio of unaccommodated nodes as the ratio $\frac{|S_2|}{|S_1|}$, where S_1 is the set of unassociated nodes with one or more ZC/ZR neighbors, and S_2 is the number of unassociated nodes with one or more ZC/ZR neighbors, some of which use the same beacon slot(s). S_1 is nonempty when the parameter combination Lm/Cm/Rm limits the admission of devices into the network. S_2 is nonempty when *in addition to* the parameter combination Lm/Cm/Rm, beacon collision further restricts the admission of devices into the network. Fig. 11 shows that the ratio of unaccommodated nodes decreases with network density. This is because as the network becomes denser, the constraint imposed by Lm/Cm/Rm has a higher impact than beacon collision. Fig. 11 also shows that the ratio of unaccommodated nodes increases with Lm. This is because as the tree becomes "taller" and hence capable of admitting more devices, beacon collision becomes a more likely cause of nodes not being able to join the network.

6 Conclusions

Traditionally, beacon schedules are chosen such that a ZR does not reuse beacon slots already claimed by its neighbors, or the neighbors of its neighbors. We observe however that beacon slots can be reused judiciously to the desirable effect of reduced packet delivery latency. Based on this idea, we have formalized a framework where we can analyze the risk of slot reuse between any two nodes. If the calculated risk is high, slot reuse is disallowed; otherwise, slot reuse is allowed. This is essentially the heart of our *ZigBee-compatible, distributed, risk-aware, probabilistic beacon scheduling algorithm*. Simulation results confirm that 1) the slot-reuse rule represented by our core algorithm is better than the slot-reuse rules espoused by the specification and prior work in the literature, and 2) centralized algorithms with complete topology information

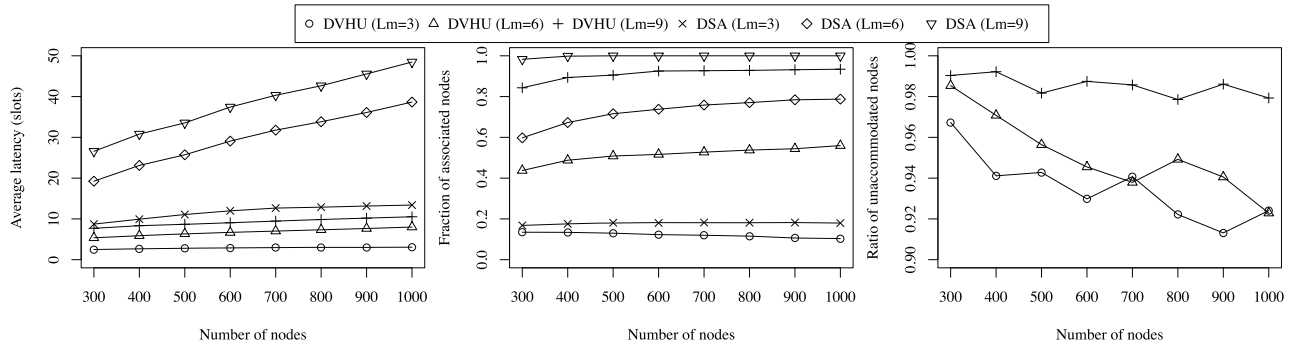


Fig. 11. Comparison of DVHU with DSA. Parameters: $r = 0.1l$, $\mathrm{Cm} = 7$, $\mathrm{Rm} = 7$, $2^{\mathrm{BO\text{-}SO}} = 128$. The slot-then-depth parent-selection policy is used.

offer hardly any advantage over their distributed counterparts that demand only local neighborhood and parenthood information. We also have proposed a new parent-selection rule as an augmentation to our core algorithm. Simulation results confirm the benefit of this rule in further reducing message latency.

ACKNOWLEDGMENTS

This work has been supported by the National Science Council, Taiwan, under contract NSC 97-2221-E-390-014. Yee Wei Law and Marimuthu Palaniswami were supported by the Australian Research Council Research Network on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), and the European Commission under contract number FP7-257992 (SmartSantander). The authors would like to thank Jiong Jin for his input.

REFERENCES

[1] *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),* IEEE, http://standards.ieee.org/getieee802/download/802.15.4-2003.pdf, 2003.

[2] ZigBee Alliance, "ZigBee-2007 Specification," Document 053474r17, http://www.zigbee.org, Jan. 2008.

[3] M.A. Lopez-Gomez, A. Florez-Lara, J.M. Jimenez-Plaza, and J.C. Tejero-Calado, "Algorithms and Methods beyond the IEEE 802.15.4 Standard for a Wireless Home Network Design and Implementation," *Proc. IEEE Int'l Conf. Sensor Networks, Ubiquitous, and Trustworthy Computing,* pp. 138-145, June 2008.

[4] A. Koubâa, A. Cunha, M. Alves, and E. Tovar, "TDBS: A Time Division Beacon Scheduling Mechanism for Zigbee Cluster-Tree Wireless Sensor Networks," *Real-Time Systems J.,* vol. 40, no. 3, pp. 321-354, Dec. 2008.

[5] V. Annamalai, S.K.S. Gupta, and L. Schwiebert, "On Tree-Based Convergecasting in Wireless Sensor Networks," *Proc. IEEE Wireless Comm. and Networking (WCNC),* pp. 1942-1947, Mar. 2003.

[6] M.-S. Pan and Y.-C. Tseng, "Quick Convergecast in ZigBee Beacon-Enabled Tree-Based Wireless Sensor Networks," *Computer Comm.,* vol. 31, pp. 999-1011, 2008.

[7] L.-W. Yeh and M.-S. Pan, "Two-Way Beacon Scheduling in ZigBee Tree-Based Wireless Sensor Networks," *Proc. IEEE Int'l Conf. Sensor Networks, Ubiquitous, and Trustworthy Computing,* pp. 130-137, June 2008.

[8] X. Chen, X. Hu, and J. Zhu, "Minimum Data Aggregation Time Problem in Wireless Sensor Networks," *Proc. Int'l Conf. Mobile Ad-Hoc and Sensor Networks,* X. Jia, J. Wu, and Y. He, eds., pp. 133-142, 2005.

[9] S. Upadhyayula, V. Annamalai, and S.K.S. Gupta, "A Low-Latency and Energy-Efficient Algorithm for Convergecast in Wireless Sensor Networks," *Proc. IEEE Global Telecomm. Conf. (GlobeCom),* pp. 3525-3530, Dec. 2003.

[10] N.-L Lai, C.-T King, and C.-H Lin, "On Maximizing the Throughput of Convergecast in Wireless Sensor Networks," *Proc. Int'l Conf. Advances in Grid and Pervasive Computing,* S. Wu, L. Yang, and T. Xu, eds., pp. 396-408, 2008.

[11] J. Zhu and X. Hu, "Improved Algorithm for Minimum Data Aggregation Time Problem in Wireless Sensor Networks," *J. Systems Science and Complexity,* vol. 21, pp. 626-636, 2008.

[12] J. Mao, Z. Wu, and X. Wu, "A TDMA Scheduling Scheme for Many-to-One Communications in Wireless Sensor Networks," *Computer Comm.,* vol. 30, pp. 863-872, 2007.

[13] L.-H. Yen, Y.W. Law, and M. Palaniswami, "Risk-Aware Beacon Scheduling for Tree-Based ZigBee/IEEE 802.15.4 Wireless Networks," *Proc. Fourth Int'l Wireless Internet Conf. (WICON '08),* 2008.

[14] I. Stojmenovic, A. Nayak, and J. Kuruvila, "Design Guidelines for Routing Protocols in Ad Hoc and Sensor Networks with a Realistic Physical Layer," *IEEE Trans. Comm.,* vol. 43, no. 3, pp. 101-106, Mar. 2005.

[15] F. Kuhn, R. Wattenhofer, and A. Zollinger, "Ad Hoc Networks beyond Unit Disk Graphs," *Wireless Networks,* pp. 715-729, 2008.

[16] G. Zhou, T. He, S. Krishnamurthy, and J.A. Stankovic, "Models and Solutions for Radio Irregularity in Wireless Sensor Networks," *ACM Trans. Sensor Networks,* vol. 2, no. 2, pp. 221-262, 2006.

[17] P. Hall, *Introduction to the Theory of Coverage Processes.* John Wiley and Sons, 1988.

[18] L.-H. Yen and Y.-M. Cheng, "Clustering Coefficient of Wireless Ad Hoc Networks and the Quantity of Hidden Terminals," *IEEE Comm. Letter,* vol. 9, no. 3, pp. 234-236, Mar. 2005.

Li-Hsing Yen received the BS, MS, and PhD degrees in computer science, all from National Chiao Tung University, Taiwan, in 1989, 1991, and 1997, respectively. He was with the Department of Computer Science and Information Engineering at Chung Hua University, Taiwan, from 1998 to 2006. He has been with the Department of Computer Science and Information Engineering, National University of Kaohsiung, Taiwan, since 2006 and is currently a full professor. His research interests include mobile computing, wireless networking, and distributed algorithms. He is a member of the IEEE.

Yee Wei Law received the BEng (1997), MEng (2002), and PhD (2005) degrees from the University of Southampton (United Kingdom), Nanyang Technological University (Singapore), and University of Twente (the Netherlands), respectively. Currently, he is a research fellow in the Department of Electrical and Electronic Engineering, the University of Melbourne, Australia. His main research interest is the security of wireless sensor networks.

Marimuthu Palaniswami received the ME degree from the Indian Institute of Science, India, the MEngSc degree from the University of Melbourne, and the PhD degree from the University of Newcastle, Australia, before rejoining the University of Melbourne. He has published more than 180 refereed papers and a huge proportion of them have appeared in prestigious IEEE journals and conferences. He was given a Foreign Specialist Award by the Ministry of Education, Japan, in recognition of his contributions to the field of machine learning. His academic excellence is recognized by several invited presentations of plenary/keynote lectures and he is a panel member for several top international conferences. He has been an associate editor for journals/transactions including *IEEE Transactions on Neural Networks.* His research interests include SVMs, sensors and sensor networks, machine learning, neural networks, pattern recognition, signal processing and control. He is a senior member of the IEEE.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.