Real-Time Syst
DOI 10.1007/s11241-008-9063-4

TDBS: a time division beacon scheduling mechanism for ZigBee cluster-tree wireless sensor networks

Anis Koubâa · André Cunha · Mário Alves · Eduardo Tovar

Abstract Synchronization is a challenging and important issue for time-sensitive Wireless Sensor Networks (WSN) since it requires a mutual spatiotemporal coordination between the nodes. In that concern, the IEEE 802.15.4/ZigBee protocols embody promising technologies for WSNs, but are still ambiguous on how to efficiently build synchronized multiple-cluster networks, specifically for the case of cluster-tree topologies. In fact, the current IEEE 802.15.4/ZigBee specifications restrict the synchronization to beacon-enabled (by the generation of periodic beacon frames) star networks, while they support multi-hop networking in mesh topologies, but with no synchronization. Even though both specifications mention the possible use of cluster-tree topologies, which combine multi-hop and synchronization features, the description on how to effectively construct such a network topology is missing. This paper tackles this issue by unveiling the ambiguities regarding the use of the cluster-tree topology and proposing a synchronization mechanism based on Time Division Beacon Scheduling (TDBS) to build cluster-tree WSNs. In addition,

This work was partially funded by FCT under the CISTER Research Unit (FCT UI 608), PLURALITY and CMU-PT projects, and by the ARTIST2/ARTISTDesign NoEs.

A. Koubâa (✉) · A. Cunha · M. Alves · E. Tovar
IPP-HURRAY! Research Group, Polytechnic Institute of Porto, School of Engineering (ISEP/IPP), Porto, Portugal
e-mail: aska@isep.ipp.pt

A. Cunha
e-mail: arec@isep.ipp.pt

M. Alves
e-mail: mjf@isep.ipp.pt

E. Tovar
e-mail: emt@isep.ipp.pt

A. Koubâa
College of Computer Science and Information Systems, Al-Imam Muhammad Ibn Saud University, Riyadh, Saudi Arabia

we propose a methodology for efficiently managing duty-cycles in every cluster, ensuring the fairest use of bandwidth resources. The feasibility of the TDBS mechanism is clearly demonstrated through an experimental test-bed based on our open-source implementation of the IEEE 802.15.4/ZigBee protocols.

Keywords Wireless sensor networks · Multi-hop synchronization · IEEE 802.15.4 · ZigBee · Real-time scheduling · Cluster-tree · Collision avoidance

1 Introduction

The joint efforts of the IEEE 802.15.4 task group (2003) and the ZigBee Alliance (2006) have ended up with the specification of a standard protocol stack for Low-Rate Wireless Personal Area Networks (LR-WPANs), an enabling technology for Wireless Sensor Networks (WSNs) (Zheng and Myung 2004; Geer 2005; Adams 2005; Culter 2005; Herzog 2005; Cunha 2007) with roughly 50% WSN market share (Zigbee-Alliance 2006). For the sake of simplicity, in the rest of this paper we will use ZigBee to denote the entire IEEE 802.15.4/ZigBee protocol stack.

ZigBee is gaining an exponentially increasing interest from both academia and industry, and is considered as an universal solution for low-cost low-power wirelessly connected monitoring and control devices (Adams 2005; Culter 2005; Herzog 2005). This interest is mainly driven by a growing number of emerging applications, including home automation and power grid telemetry (as the current principal commercial targets of the ZigBee Alliance), health care monitoring, industrial automation, environmental monitoring, structural monitoring and surveillance. WSNs represent the new generation of network infrastructures for enabling such large-scale distributed embedded systems.

The potential of ZigBee, even though technology is still immature and, to a certain extent, still costly, is closely related to the objectives for which it was designed (IEEE-TG15.4 2003; Zigbee-Alliance 2006) and to its flexibility to fit different network and application requirements. While it was designed for low-cost wireless devices (such as wireless sensor nodes), ZigBee is able to provide low power consumption and real-time guarantees. However, the benefit gained from these features typically depends on the configuration of the Medium Access Control (MAC) sub-layer, whether operating in beacon-enabled (with synchronization) or in non beacon-enabled (without synchronization) modes. At a first glance, the non beacon-enabled mode may be an interesting solution for large-scale WSNs by adopting ZigBee mesh topologies. However, only in the beacon-enabled mode it is possible to dynamically adapt low duty-cycles (from 100%) down to 0.006%, which is interesting for most WSN applications—where energy efficiency is one of the main concerns. In addition, the beacon-enabled mode also offers some real-time guarantees by means of the Guaranteed Time Slot (GTS) mechanism, an attractive feature for time-sensitive WSN applications. On the other side, the non beacon-enabled mode does not provide any of these features, but it has the advantage of lower complexity and high routing redundancy scalability when compared to the beacon-enabled mode, since the former does not require any synchronization. Note that in the context of ZigBee, synchronization

means that a central device, called the ZigBee Coordinator (ZC)—also referred to as PAN Coordinator, periodically transmits beacon frames to its neighbour child nodes, in star network topologies (1 cluster). For the case of multiple cluster networks, beacon frames are broadcasted throughout the network via the ZigBee Routers (ZRs) according to a cluster-tree topology. Each ZR synchronizes the (child) nodes in their cluster. The bottom line of this reasoning is the follows: WSN applications with stringent energy and/or delay requirements must operate in the beacon-enabled mode.

It happens, however, that the beacon-enabled mode suffers from not offering scalability since, inherently to its operational behaviour, it was limited to star topologies, while solutions for beacon/superframe scheduling in cluster-tree topologies (such as the proposed Time Division Beacon Scheduling (TDBS)) did not exist. In a star-based network operating in beacon-enabled mode, beacon frames are periodically transmitted by the ZC, for synchronizing the nodes in its vicinity. As a consequence, the network coverage is limited to the transmission range of the ZC, which restricts the geographical coverage of the network. This is particularly unsuitable for WSNs, which most often spread out over large regions. This leads to contradictory requirements between supporting scalability at the cost of energy consumption (mesh) and delay guarantees, and vice-versa (star). It would be more appropriate if both features (synchronization and scalability) could be simultaneously supported in the same network.

In that direction, the ZigBee specification also defines the concept of cluster-tree topology. A cluster-tree network is formed by several coordinators (the ZC and the ZRs) that periodically send beacon frames to the nodes in their clusters, providing them synchronization and cluster management services. However, from our perception of the IEEE 802.15.4/ZigBee specifications, and based on some interactions with some members of the ZigBee Alliance, there is no clear description on how the cluster-tree model (proposed in IEEE-TG15.4 2003, Sect. 5.2.1.2) can be implemented. The available information regarding this model gives only a vague overview on how the cluster-tree network should operate, and some details on the tree routing algorithm (Zigbee-Alliance 2006). Moreover, the interaction between the Data Link and Network Layers is also missing.

More specifically, in the cluster-tree model, each cluster is managed by one ZR, which generates periodic beacon frames to synchronize its child nodes (that belong to its cluster). In this case, if these periodic beacon frames are sent by the ZRs in a non-organized fashion (with no particular schedule), they will collide (either with each other or with data frames). In fact, in case of beacon frame collisions, nodes will lose synchronization with their coordinators, and consequently with the network, preventing them to communicate. As a consequence, beacon frame scheduling mechanisms must be defined to avoid beacon frame collisions in ZigBee cluster-tree networks. The problem that we tackle in this paper can be roughly formulated as follows:

Synchronization in a ZigBee cluster-tree network: given an IEEE 802.15.4/ZigBee network with several coordinators generating periodic beacon frames and organized in a cluster-tree topology, how to schedule the generation time offsets of beacon frames issued from different coordinators to completely avoid beacon frame collisions with each other and with data frames.

In this paper, we propose a solution to overcome this problem based on a collision-free beacon frame scheduling algorithm. To our best knowledge, the beacon frame

scheduling problem has not been resolved neither by the IEEE 802.15.4/ZigBee standardization groups nor by any previous research work.

Contributions of the paper In this paper, we propose a scheduling mechanism (TDBS) for building a ZigBee cluster-tree WSN based on a time division approach. Importantly, the TDBS mechanism can easily be integrated in the IEEE 802.15.4/ZigBee protocol stack with only minor add-ons. Another motivation is that in Koubaa et al. (2006) we have tackled the worst-case dimensioning of cluster-tree networks under this approach. Thus, by engineering cluster-tree networks according to TDBS, it will be possible to experimentally assess and validate the theoretical results obtained in Koubaa et al. (2006).

The main contributions of this paper are the following:

First, we identify and analyze the beacon frame collision problem (Sect. 3), and the different approaches proposed in Task Group 15.4b: (http://grouper.ieee.org/groups/802/15/pub/TG4b.html) to avoid it (Sect. 4).

Second, we propose a beacon frame scheduling mechanism (TDBS) based on the time division approach to build a synchronized multi-hop cluster-tree WSN (Sect. 5).

Third, we present a duty-cycle management methodology for an efficient utilization of bandwidth resources in the cluster-tree network (Sect. 6).

Fourth, we demonstrate the feasibility of the TDBS mechanism through an experimental test-bed (Sect. 7).

2 Related work

Clustering and multi-hop network synchronization are common problems in WSNs, and have been addressed in many research works (e.g. Rowe et al. 2006; Heinzelman et al. 2000; Gupta and Younis 2003; Kottapalli et al. 2003; Caccamo and Zhang 2002; Rajendran et al. 2003). In Rowe et al. (2006), the authors have presented the RT-Link protocol, which provides a centralized synchronization scheme based on time sync pulses sent by a global clock for indoor and outdoor fixed devices. These sync pulses indicate the beginning of each time-slotted cycle and the first frame being sent. This protocol is similar to the IEEE 802.15.4 protocol since it also uses a superframe structure with contention-based and contention-free periods. The authors also proposed a centralized mechanism for dynamic slot re-use and assignment in multi-hop networks, to reduce delays. The use of a global synchronization clock, however, may not be practical for some sensor network applications, especially for synchronization in large-scale networks.

In Caccamo and Zhang (2002), the authors have proposed Implicit-Earliest Deadline First (Implicit-EDF), which assumes that all nodes in each cluster hear each other and that a table containing the characteristics of all the periodic traffic of the cluster has to be transmitted to the nodes in advance. This assumption is uncommon in most of WSN applications.

In Heinzelman et al. (2000), the authors proposed LEACH, a clustering-based protocol using a randomized rotation and selection of cluster-heads to optimize energy consumption. After the random selection of cluster-heads, the other nodes decide to which cluster they belong, and inform the corresponding cluster-head (using

CSMA/CA) of their decision. After the reception of all join requests, cluster-heads compute a TDMA (Time Division Multiple Access) schedule according to the number of nodes in their cluster. This schedule is broadcasted back to the node in the cluster. Inter-cluster interference is mitigated using different CDMA (Code Division Multiple Access) codes in each cluster. The clustering and synchronization approach in Heinzelman et al. (2000) differs from the ZigBee approach in three aspects, which turns our problem quite different. First, concerning clustering in ZigBee networks, coordinators (or cluster-heads) are fixed (do not change during run-time). Second, the synchronization is not made using a TDMA schedule, but by means of periodic beacon frame transmissions, which has the advantage of higher flexibility (TDMA is not scalable and is vulnerable to dynamic network changes). Finally, ZigBee does not allow the use of CDMA to avoid inter-cluster interferences, which leads to collisions between beacon and data frames issued in different clusters. In our case, a node that experiences (repeated) collisions of beacon frames will inevitably lose synchronization. Hence, there is a need to schedule different beacon frames from different coordinators to avoid beacon frame losses that lead to undesirable synchronization problems.

Triggered by this problem, the Task Group 15.4b (http://grouper.ieee.org/groups/802/15/pub/TG4b.html) proposed for discussion some basic approaches for avoiding beacon frame collisions that may be adopted in the upcoming amendments to the standard. A first approach, called the Beacon-Only Period approach, consists in having a time window at the beginning of each superframe reserved for beacon frame transmissions. The second approach, based on time division, proposes that beacon frames of a given cluster are sent during the inactivity periods of the other clusters. However, the algorithms showing how to schedule beacon frame transmission in a collision-free fashion are not presented. More specifically, the approaches proposed by the Task Group 15.4b show how to extend the standard to create offsets in the beacon transmit time and to take beacon frame scheduling into account, but how to choose the time offsets of different beacons is not addressed. Surprisingly, the approaches discussed in the Task Group 15.4b were not (fully included) in the new versions of the standard IEEE 802.15.4b (2006) and ZigBee specification (DEC/2006). We elaborate further on these aspects later on in Sect. 4.

Some recent research works have addressed the feasibility analysis and the design of ZigBee cluster-tree networks. In Mirza et al. (2005) the authors have proposed a centralized mechanism for scheduling beacon frames to reduce energy consumption of energy-critical nodes for data aggregation-based applications. However, in that approach there are no guarantees on keeping synchronization due to possible beacon frame collisions. In Ha et al. (2005) the feasibility of ZigBee multiple cluster (cluster-tree) beacon-enabled networks has been analyzed under the assumption of clusters working at full duty cycle (beacon order equals superframe order). The authors conclude, with some empirical guidelines, that cluster-tree networks are feasible, but they do not provide any solution to the beacon scheduling problem in order to avoid beacon collision. In fact, they only analyse the probability of failed transmissions of both beacon transmissions and association procedures.

3 IEEE 802.15.4/ZigBee protocols overview

In ZigBee networks there are 3 types of devices: (1) ZigBee Coordinator (ZC); (2) ZigBee Router (ZR) and (3) ZigBee End Device (ZED). A ZC is a Full Function Device (FFD). There is one for each ZigBee Network that initiates and configures the network formation. A ZC acts as an IEEE 802.15.4 PAN Coordinator and also as a ZigBee Router (ZR) once the network is formed. A ZigBee Router (ZR) is also a FFD. It is associated with the ZC or with a previously associated ZR. It acts as an IEEE 802.15.4 Coordinator and participates in multi-hop routing. Finally, a ZED does not allow other devices to associate with it. A ZED has no routing functionality and may implement a reduced subset of the protocol stack (RFD—Reduced Function Device).

The IEEE 802.15.4 Medium Access Control (MAC) protocol supports two operational modes that may be selected by the ZC: (1) the non beacon-enabled mode, in which the MAC is simply ruled by non-slotted CSMA/CA and (2) the beacon-enabled mode, in which beacons are periodically sent by the ZC (and by the ZRs, if they exist) to synchronize nodes that are associated with it, and for network management purposes.

ZigBee enables three network topologies (Fig. 1): star; mesh and cluster-tree. In all cases, a unique node operates as a ZC. The ZC chooses a Personal Area Network (PAN) identifier, which must not be used by any other ZigBee network in the vicinity.

In the *star topology* (Fig. 1a), the communication paradigm is centralized, i.e. each device (FFD or RFD) joining the network and willing to communicate with other devices must send the data to the ZC, which dispatches it to the adequate destination node. The star topology is not adequate for most WSN due to the lack of scalability. This lack of scalability does not result from the allowable number of nodes (maximum addressing space of 65535) but from the limitation in terms of covered region, since all nodes in the cluster must be within the radio coverage of the ZC. Star networks can operate both in beacon-enabled and non beacon-enabled modes.

In the *mesh topology* (Fig. 1b) the communication paradigm is decentralized; each node can directly communicate with any other node within its radio range. The mesh topology enables enhanced networking flexibility, but it induces an additional complexity for providing end-to-end connectivity between all nodes in the network. Basically, the mesh topology operates in an ad-hoc fashion and allows multi-hop routing from any node to any other node. The mesh topology may be more energy efficient than the star topology since communications do not rely on one particular node, but does not allow efficient duty-cycle management due to the lack of synchronization (only operates in non beacon-enabled mode), thus leading to limited network lifetime.

The *cluster-tree topology* (Fig. 1c) is a special case of a mesh network where there is a single routing path between any pair of nodes and there is a distributed synchronization mechanism (operates in the beacon-enabled mode). There is a unique ZC in the network and one ZR per cluster. Any FFD can act as a ZR and provide synchronization services to its child nodes (ZEDs or ZRs). In cluster-tree networks, duty-cycles can be dynamically managed in a per-cluster basis and it is possible to carry out the worst-case network dimensioning, both in terms of worst-case buffer

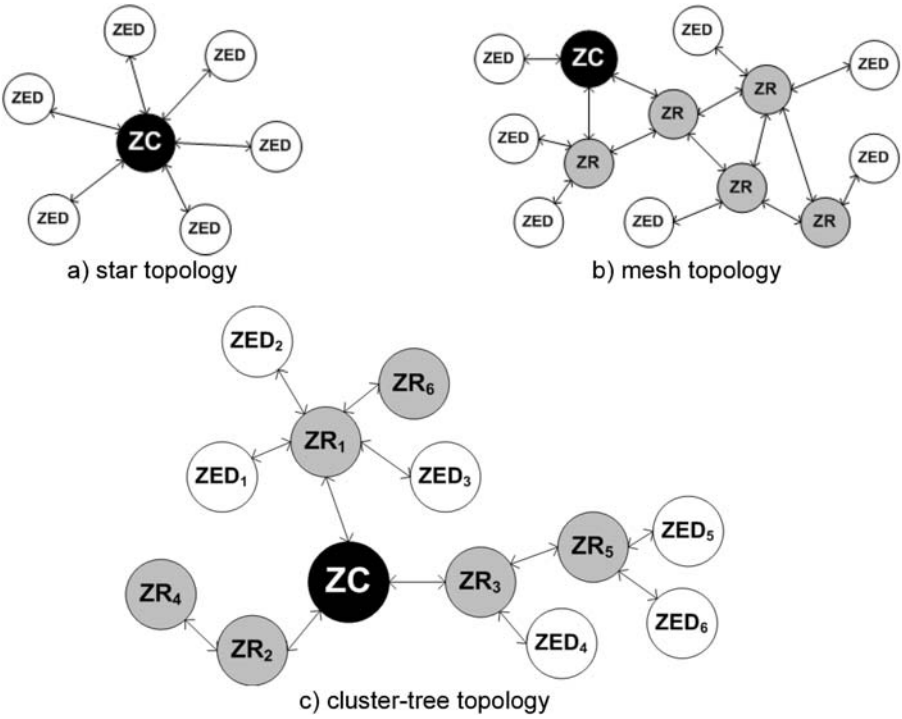


Fig. 1 IEEE 802.15.4 network topologies

occupation and message end-to-end delays (Koubaa et al. 2006; Jurčík et al. 2008). In this paper, we focus on the cluster-tree model due to its scalability, energy-efficiency and real-time properties.

In beacon-enabled mode, the ZC defines a superframe structure (refer to Fig. 2), which is constructed based on the Beacon Interval (*BI*), which defines the time between two consecutive beacon frames and on the Superframe Duration (*SD*), which defines the active portion of the *BI*, and is divided into 16 equally-sized time slots during which frame transmissions are allowed. Optionally, an inactive period is defined if $BI > SD$. During the inactive period (if it exists), all nodes may enter into a sleep mode (to save energy).

BI and *SD* are determined by two parameters, the Beacon Order (*BO*) and the Superframe Order (*SO*), as follows:

$$\left.\begin{aligned} BI &= aBaseSuperframeDuration \cdot 2^{BO} \\ SD &= aBaseSuperframeDuration \cdot 2^{SO} \end{aligned}\right\} \quad \text{for } 0 \leq SO \leq BO \leq 14. \tag{1}$$

$aBaseSuperframeDuration = 15.36$ ms (assuming 250 kbps in the 2.4 GHz frequency band) denotes the minimum duration of the superframe, corresponding to $SO = 0$.

During the *SD*, nodes compete for medium access using slotted CSMA/CA in the Contention Access Period (CAP). For applications requiring guaranteed bandwidth, IEEE 802.15.4 enables the definition of a Contention-Free Period (CFP) within the SD, by the allocation of Guaranteed Time Slots (GTS).

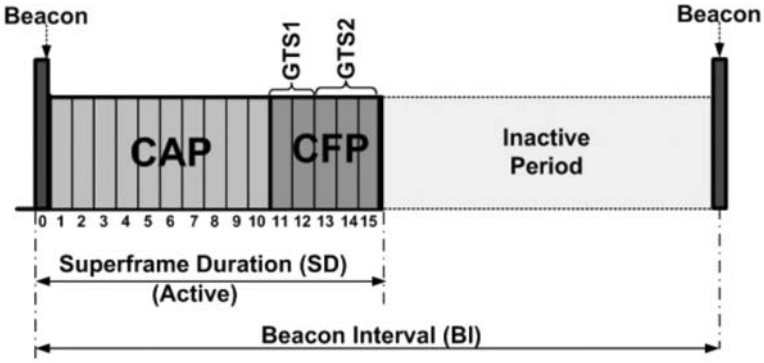


Fig. 2 Superframe structure (IEEE-TG15.4 2003)

Low duty-cycles can be configured by setting small values of the superframe order (*SO*) as compared to beacon order (*BO*), resulting in greater sleep (inactive) periods (in Fig. 2, $BO = SO + 1$, so the nodes may sleep half of the time). The advantage of this synchronization with periodic beacon frame transmissions from the ZigBee coordinator is that all nodes (in a cluster) wake up and enter sleep mode at the same time. However, as discussed earlier, using this synchronization scheme in a cluster-tree network with multiple coordinators sending beacon frames, each with its own beacon interval, is a challenging problem due to beacon frame collisions. This is addressed in this paper.

4 ZigBee cluster-tree network model and the beacon collision problem

The beacon frame collision problem in cluster-tree ZigBee WPANs has been addressed as Request for Comments (RFC) in the Task Group 15.4b (http://grouper.ieee.org/groups/802/15/pub/TG4b.html). In this section we identify the cluster-tree network model and analyze the different types of beacon frame collision conflicts identified by that task group.

4.1 Network model

In this paper, we consider a cluster-tree network as exemplified in Fig. 1c. The network is identified by the ZigBee Coordinator (ZC), which is unique. The ZigBee Coordinator and the ZigBee Routers send periodic beacon frames to synchronize the nodes in their clusters. Throughout this paper, we interchangeably use ZigBee Router and Coordinator, and both are denoted by ZR. Hence, each coordinator ZRi acts as a cluster-head of cluster i for all its child nodes (that are associated to the network through it), and as a consequence, will send periodic beacon frames to keep them synchronized. The cluster-tree is formed by several parent-to-child associations between ZigBee Routers until a certain depth. In Fig. 1c, for instance, ZR_3 is a parent coordinator of ZR_5 and a child coordinator of the ZigBee Coordinator, considered as the root of the tree.

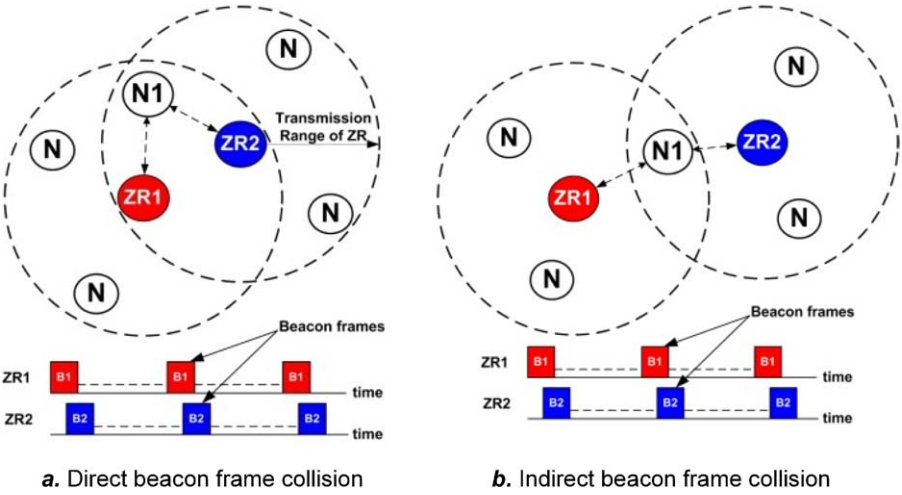


Fig. 3 The beacon frame collision problem

It is easy to notice that sending periodic beacon frames without special care on timing issues may result in beacon frame collisions in some nodes that are in the range of more than one coordinator. The Task Group 15.4b has identified two types of collisions: (1) direct beacon frame collisions and (2) indirect beacon frame collisions. Both types are briefly explained next.

4.2 Direct beacon frame collisions

Direct beacon frame collisions occur when two or more coordinators are in the transmission range of each other (direct neighbours or parent-to-child relation) and send their beacon frames at approximately the same time, as illustrated in Fig. 3a. In that example, assuming that node N1 is a child of ZR1, which sends its beacon frame at approximately the same time as ZR2, node N1 loses its synchronization with its parent ZR1 due to the collision of beacon frames from ZR1 and ZR2.

4.3 Indirect beacon frame collisions

Indirect beacon frame collisions occur when two or more coordinators cannot hear each other, but have overlapped transmission ranges (indirect neighbours) and send their beacon frames at approximately the same time (hidden-node problem), as shown in Fig. 3b. Two undesirable situations may occur. The first is when a child node N1, already associated to a parent coordinator ZR1, and the coordinator ZR2 start both sending their beacon frames at almost the same time, therefore resulting in the loss of the child node synchronization with its parent. The second case is when two coordinators (in the example ZR1 and ZR2) belonging to the same PAN cannot hear each other and send their beacon frames at the same time. In this case, any node in the overlapping transmission range of both coordinators will be prevented from joining the network. Note that collisions between data and beacon frames may also happen,

for example when a coordinator sends its periodic beacon frame during the active period of an adjacent cluster. Hence, this problem must also be dealt with.

5 Basic approaches of the Task Group 15.4b for beacon collision avoidance

Since no mechanism to avoid beacon frame collisions is considered in the current IEEE 802.15.4 standard, some proposals have been discussed in Task Group 15.4b. These were proposed as pattern ideas to trigger the design of solutions to the beacon frame collision problem. In what follows, we outline these proposals.

5.1 Direct beacon frame collision avoidance

Two approaches were proposed to avoid the direct beacon frame collision problem (Fig. 4).

5.1.1 The time division approach

In this approach, time is divided such that beacon frames and the superframe duration of a given cluster are scheduled in the inactive period of its neighbour cluster, as shown in Fig. 4a. The idea is that each coordinator uses a starting time *Beacon_Tx_Offset* to transmit its beacon frames, that must be different from the starting times of its neighbour coordinators and their parents. This approach requires that a coordinator wakes up both in its active period and in its parent's active period, so that it can route upstream and downstream.

The limitations of this approach are: (1) it constraints the duty-cycles, since they will be dependent on the number of interfering coordinators (which must operate in different time windows); (2) direct communication between neighbour clusters is not possible, since overlapping clusters must operate in time windows. The density of devices that can be supported is inversely proportional to the ratio of the beacon order and superframe orders, assuming that all *BOs* and *SOs* are equal for all clusters. References to this approach are introduced in the ZigBee standard (Zigbee-Alliance 2006).

Observe that *Beacon_Tx_Offset* must be chosen adequately, not only to avoid beacon frame collisions, but also to enable efficient utilization of inactive periods, thus maximizing the number of clusters in the same network. This problem is more challenging when the superframe orders and beacon orders are different from one cluster to another. This issue is addressed in Sect. 6.

5.1.2 The beacon-only period approach

In this approach, a time window, denoted as Beacon-Only Period, is reserved at the beginning of each superframe for the transmission of beacon frames in a contention-free fashion (Fig. 4b). Each coordinator chooses a sending time offset by selecting a contention-free time slot (CFTS) such that its beacon frame does not collide with beacon frames sent by its neighbour clusters. The advantage of this approach as compared to the previous one is that the active periods of the different clusters start at the

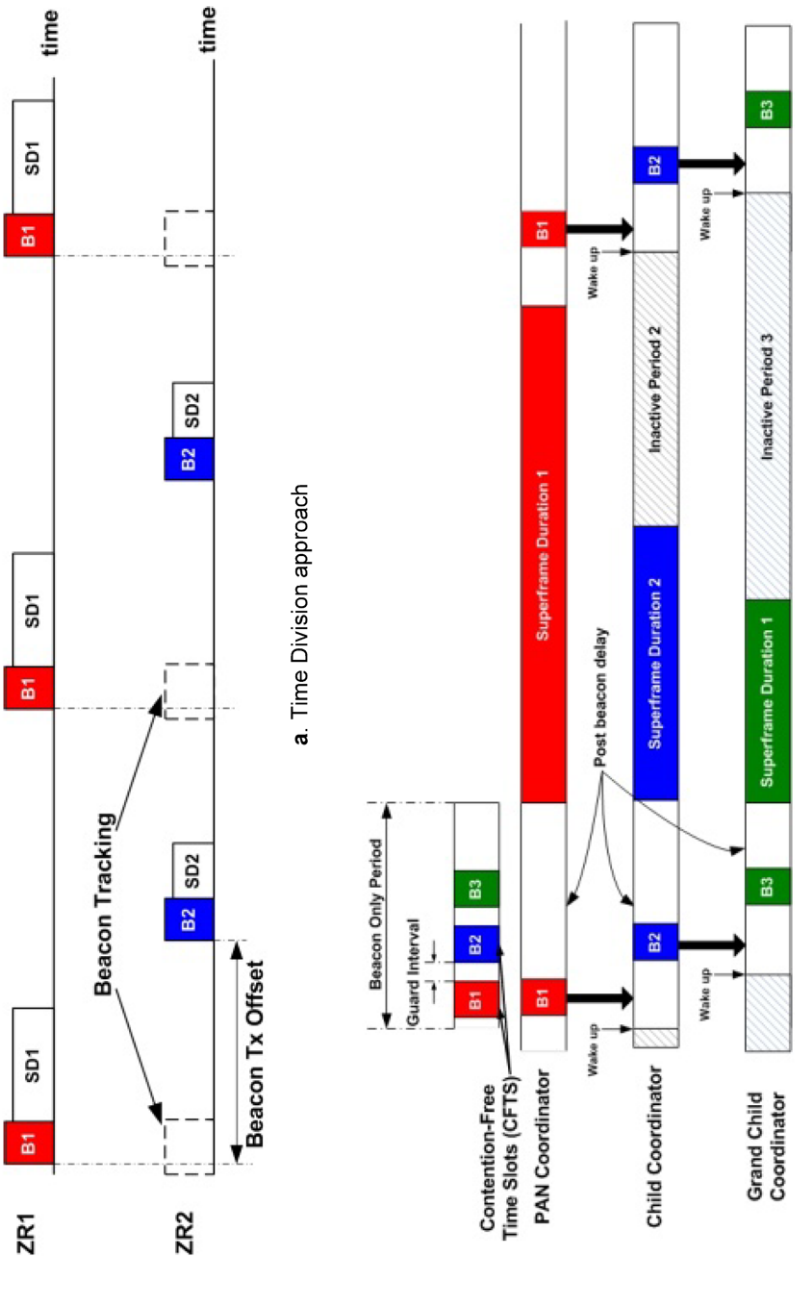


Fig. 4 Beacon frame collision avoidance approach

same time, thus direct communication between neighbor nodes is possible, and there is no constraint on the duty-cycle.

The main complexity of this approach is the dimensioning of the duration of the beacon-only period for a given network topology. This duration depends on the number of nodes in the network, their parent-child relationship and also the scheduling mechanism used to allocate the CFTS to each coordinator. Additionally, the GTS mechanism cannot be implemented (at least in accordance to the specification), since transmission from nodes belonging to different clusters may collide. Thus, transmissions are only allowed during the CAP, which would be shared by different clusters. Importantly, oppositely to the time division approach, the beacon-only period approach implies a non-negligible change to the standard protocol.

5.2 Discussion

We have presented the two approaches proposed by Task Group 15.4b to avoid direct beacon frame collisions. Note that these approaches do not include the algorithms to schedule beacon frames transmission. For the time division approach, the organization of the different superframe durations must be evaluated with care, to maximize the number of clusters in the network. For the other approach, the beacon-only period must be efficiently dimensioned.

In the next sections, we explore the time division beacon scheduling approach, namely proposing mechanisms for collision-free beacon/superframe scheduling and for efficiently computing the duty-cycles in a cluster-tree network, as well as presenting an experimental test bed.

6 The time division beacon frame scheduling (TDBS) mechanism

6.1 Problem formulation

Let us consider an IEEE 802.15.4/ZigBee network as presented in Fig. 1c with a set of N coordinators (including the ZC) $\{ZR_i = (SD_i, BI_i)\}$, for $1 \leq i \leq N$ that generate periodic beacon frames with a given superframe order SO_i and beacon order BO_i. SD_i and BI_i denote the superframe duration and the beacon interval of the ith coordinator ZR_i, respectively. The problem is how to organize the beacon frames of the different coordinators to avoid collisions with other beacon and data frames, using the time division approach.

The most intuitive idea is to organize beacon frame transmissions in a non-overlapping way such that no beacon frame will collide with another even if coordinators are in direct or indirect neighbourhood (refer to Sect. 4). In addition, to avoid collisions with data frames, a beacon frame must not be sent during the superframe duration of another coordinator. Thus, the beacon frame scheduling problem comes back to a superframe scheduling problem, since each superframe starts with a beacon frame.

At a first glance, this problem can be considered as a non-preemptive scheduling of a set of periodic tasks, where the execution time of a task is equal to the superframe

duration, and the period is equal to the beacon interval. However, the additional restriction in the superframe scheduling problem is that consecutive instances of SD must be separated by exactly one beacon interval (BI).

In what follows, we propose a superframe scheduling algorithm.

6.2 Superframe duration scheduling (SDS) algorithm for the time division approach

In case of equal superframe durations, the superframe scheduling problem is somewhat similar to the *pinwheel scheduling problem* presented in Holte et al. (1989). The pinwheel problem consists in finding for a set of positive integer $A = (a_1, \ldots, a_n)$ a cyclic schedule of indices $j \in (1, 2, \ldots, n)$ such that there is at least one index j within any interval of a_j slots. By analogy to our problem, given a set of beacon intervals $A = (BI_1, \ldots, BI_N)$, the problem is to find a cyclic schedule of superframe durations such that there is at least one SD_i in each BI_i. In addition to the pinwheel problem, the distance between two consecutive instances of SD_i must be equal to BI_i. In this paper, we propose a general result for the scheduling problem for different and equal superframe durations.

Theorem T1 *Let*

$$C_M = \left\{ \begin{array}{l} A | A = \{a_1, \ldots, a_N\} \text{ where } i < j \\ \Rightarrow a_i \text{ divides } a_j \text{ and } \sum_{i=1}^{N} 1/a_i \leq 1 \end{array} \right\}$$

For an instance $A \in C_M$, if a cyclic schedule exists, then the least common multiple of all integers, $LCM(a_1, a_2, \ldots, a_n) = \max_{1 \leq i \leq N}(a_i)$, is the minimum cycle length.

Proof The proof is made by contradiction. Assume that a cyclic schedule exists for an instance $A \in C_M$ of the pinwheel problem. Since $\forall i < j \Rightarrow a_i$ divide a_j, then $\forall i < j$, it exists an integer k_{ij} such that $a_j = k_{ij} \cdot a_i$ (harmonic integers). Then, we have $LCM(a_1, a_2, \ldots, a_n) = \max(a_i)$ for $1 \leq i \leq N$.

Assume that the minimum cycle length is different from $LCM(a_1, a_2, \ldots, a_n)$. Then, since $LCM(a_1, a_2, \ldots, a_n)$ is not a cycle length, there is a time slot n that contains the integer a_i such that the $(n + LCM(a_1, a_2, \ldots, a_n))$th time slot does not contain a_i. Since $LCM(a_1, a_2, \ldots, a_n)$ is a multiple of a_i, it directly implies that the set is not schedulable, which is absurd. □

According to Theorem T1, the superframe duration scheduling decision problem is PSPACE (by analogy to the pinwheel problem, which is also shown to be PSPACE). Thus, we propose the Superframe Duration Scheduling (SDS) algorithm, which performs the schedulability analysis of a set of superframes with different durations and beacon intervals, and provides a schedule if the set is schedulable. The algorithm also holds for equal superframe durations.

Let us consider a set of N coordinators $\{ZR_i = (SD_i, BI_i)\}$, for $1 \leq i \leq N$ with different superframe durations.

```
The Superframe Duration Scheduling Algorithm
A = {2^BOi}1≤i≤N the set of beacon intervals in the
cluster-tree network
BImin = 2^BOmin be the minimum beacon interval
organize the set A = {2^BOi}1≤i≤N in the increasing
order of BOi such that
if (for a given i, j we have BIi = BIj ) then
        if ( SDi ≥ SDj ) then put BIi before BIj in the set A
            else put BIj before BIi in the set A
Consider the slotted time line of length BImaj where
the size of a slot is equal to min(SDi)1≤i≤N
for (each element i in the organized set A) do {
   search the first available consecutive time slots with
  a length at least equal to SDi
  write (i) in SDi consecutive time slot starting from the
   first available time slot
repeat
   if (write( i) in SDi consecutive time slots after each
                                        BIi interval) = false)
   then return("the set is not schedulable")
until (end Major Cycle).                          }
Return ("the set is schedulable")
```

Fig. 5 The superframe duration scheduling algorithm

First, for being schedulable, it is necessary to satisfy that the sum of the duty-cycles is lower than 1, which gives the following necessary condition.

$$\sum_{i=1}^{N} DC_i = \sum_{i=1}^{N} \frac{SD_i}{BI_i} \leq 1. \tag{2}$$

Based on Theorem T1, it is sufficient to analyze the schedulability of the superframe durations in a hyper-period equal to:

$$\overline{BI}_{maj} = LCM(2^{BO_1}, 2^{BO_2}, \ldots, 2^{BO_n}) = \max_{1 \leq i \leq N} (2^{BO_i}). \tag{3}$$

This hyper-period is referred to as major cycle. The minimum beacon interval is referred to as minor cycle.

The SDS algorithm is presented in Fig. 5.

First, we denote as $A = \{2^{BO_i}\}$ for $1 \leq i \leq N$ the set of beacon interval of all coordinators in the cluster-tree network. Let $\overline{BI}_{min} = 2^{BO_{\min}}$ be the minimum beacon interval, called the minor cycle. Then, the set $A = \{\overline{BI}_i = 2^{BO_i}\}$ for $1 \leq i \leq N$ is

Table 1 Network configuration

Coordinator	*SD*	*BI*
ZR1	4	16
ZR2	1	8
ZR3	2	16
ZR4	1	32
ZR5	4	32
ZR6	2	16

organized in an increasing order such that if it exists i, j where $\overline{BI}_i = \overline{BI}_j$, then put $\overline{BI}_i, \overline{BI}_j$ in the set A in the decreasing order of their superframe durations. Hence, if $\overline{SD}_i \geq \overline{SD}_j$, then put $\overline{BI}_i$ before $\overline{BI}_j$ in the set A. Let us define a slotted time line of a length equal to the major cycle $\overline{BI}_{maj}$ and where the size of each slot is equal to the minimum superframe duration SD (time unit corresponding to $SO = 0$). For each element i in A, schedule the superframe duration SD_i by searching the first available time slot in the slotted timeline, and write the index i in SD_i consecutive time slots. This operation is repeated for all consecutive time slots located after each BI_i interval, until reaching the end of the major cycle. This algorithm returns "not schedulable" if a given superframe duration cannot find periodic free time slots in the major cycle, otherwise the set is considered as schedulable.

6.3 Illustrative example of the SDS algorithm

To illustrate the SDS algorithm, let us consider the example presented in Table 1. In Sects. 6.3 and 6.4, each time unit corresponds to a base superframe duration (i.e. $SO = 0$).

The SDS algorithm applied to the example in Table 1 is presented in Fig. 6, where the lines are single steps of the algorithm and the last line presents the final schedule. Observe that in this set, the major cycle corresponds to $\overline{BI}_{maj} = 32$ and the minor cycle corresponds to $\overline{BI}_{min} = 8$. Based on Line 04 in Fig. 5, the set of coordinators is arranged as (ZR2, ZR1, ZR3, ZR6, ZR4, ZR5) corresponding to the set $A = \{8, 16, 16, 16, 32, 32\}$. We consider the slotted timeline of length 32 time units (major cycle).

Based on Line 11 of the algorithm (Fig. 5), for each element in A, we place the first instance of the superframe duration of the corresponding coordinator in the first available time slots such that the superframe duration can fit without overlapping with other superframe durations. For instance, the first instance of the superframe duration of Coordinator ZR2 is placed in the first time slot, and the subsequent instances are placed at a distance equal to a multiple of 8 time slots from the first instance. Then, the first instance of the superframe duration of ZR1 is placed just after the first superframe duration of ZR2 (time slot 2). The subsequent instances of ZR1 are placed at a distance equal to a multiple of 16 time slots from the first instance, and so on. Observe in timeline (7) of Fig. 6 that this set is schedulable since all superframe durations are periodic and are not overlapping within the major cycle.

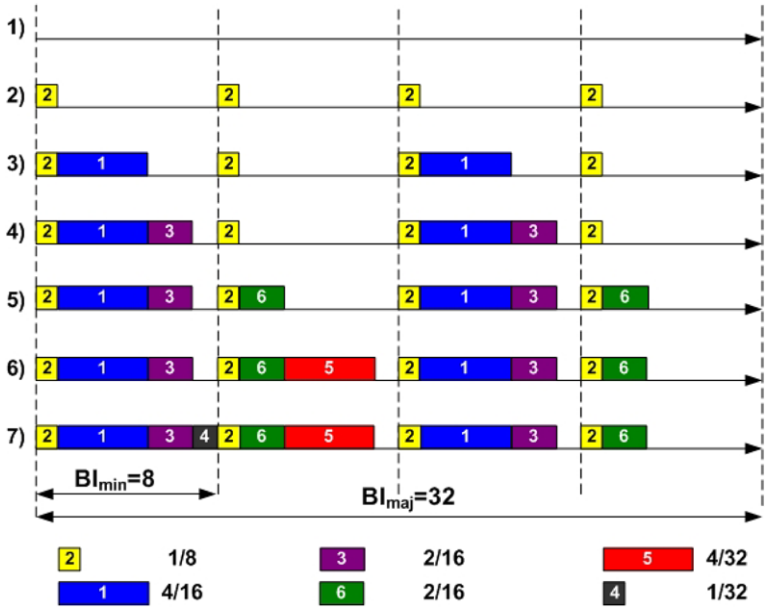


Fig. 6 Timing diagram of the SDS algorithm—example scenario

6.4 Superframe scheduling with cluster grouping

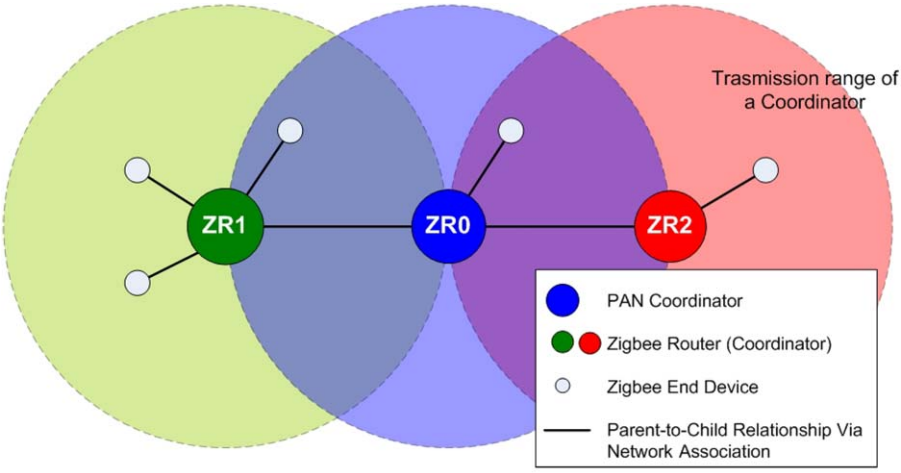
In this section, we extend the time division approach to optimize the superframe scheduling algorithm in large-scale networks. Observe that coordinators that are far enough such that their transmission ranges do not overlap, can transmit their beacon frames simultaneously (share the same time window) without facing the direct and indirect beacon frame collision problems.

To give an intuitive illustration of the approach, we propose an example with 3 coordinators located as presented in Fig. 7 and the (*BI*, *SD*) pairs as presented in Table 2.

Figure 7a shows that beacon frames from ZR0, ZR1 and ZR2 could collide since ZR0 has overlapping transmission ranges with both ZR1 and ZR2. According to (2), note that it is not possible to schedule the superframes of the three coordinators because the total duty-cycle is greater than 1 ($[0.5 + .05 + 0.5 = 1.5] > 1$). However, observe that coordinators ZR1 and ZR2 can send their beacon frames at the same time, since they are neither in direct nor in indirect neighbourhood (no overlapping transmission ranges). Thus, it possible that ZR0 sends its beacon frame followed by coordinators ZR1 and ZR2, which may send their beacon frames simultaneously. In this case, no beacon frame collision occurs, and thus the coordinator set becomes schedulable, as presented in Fig. 7.

In what follows, we propose a method to group coordinators that can send their beacon frames simultaneously.

Let us consider that each coordinator has a maximum transmission range defined by a circle of radius r. Two non-overlapping coordinators means that they are geographically separated by a distance at least equal to $2 \cdot r$, thus both can send beacon



a. The geographic distribution of the nodes in the network

b. Superframe duration scheduling with coordinator grouping

Fig. 7 Coordinator grouping example

Table 2 Network configuration

Coordinator	*SD*	*BI*
ZR0	1	2
ZR1	1	2
ZR2	1	2

frames at the same time. Hence, the problem can be considered as the vertex colouring problem of graph theory (Diestel 2005) where vertices are the coordinators and an edge is a link between two coordinators that are at least $2 \cdot r$ away. The vertex colouring algorithm can be implemented in the ZigBee Coordinator, which is assumed to know the location of all coordinators in the network, to perform group assignments and to send back the grouping information to the nodes. After processing the vertex colouring algorithm, each coordinator with the same colour belongs to the same group and all coordinators belonging to a group can send beacon frames simultaneously. Note that, in this paper, we do not address the issue of how node location is known by the coordinator, but we may consider a static topology where all the locations are know *apriori*, or use already existing location discovery mechanisms.

This grouping strategy has the advantage of permitting to find a schedule for a set of coordinators whose sum of duty-cycles is greater than one (as presented in the previous example).

7 Efficient duty-cycle management in ZigBee cluster-tree networks

In this section, we propose a methodology for assigning the adequate duty-cycles to each coordinator/cluster in the cluster-tree network, to ensure an efficient utilization of bandwidth.

7.1 Problem statement

When deploying a ZigBee cluster-tree WSN using the time division approach *without cluster grouping*, each coordinator will activate its cluster during a separate time window with a specific duty-cycle. If the duty-cycle is randomly assigned, it may lead to undesirable situations. In fact, if a coordinator has unnecessarily been assigned a high duty-cycle, it may prevent other coordinators clusters to increase their bandwidth requirements or new coordinator/clusters to join the network. On the other hand, it has been shown that each coordinator can guarantee a certain amount of bandwidth using the GTS mechanism and this bandwidth is proportional to the duty-cycle of this coordinator, as shown in Cunha (2007), Koubaa et al. (2006). Thus, if a coordinator has been assigned a low duty-cycle, it may suffer from congestion if the cluster traffic requirements are greater than the available bandwidth.

As a consequence, the adequate assignment of the duty-cycle in each coordinator is necessary to ensure an optimal distribution of the bandwidth resources among all coordinators in the cluster-tree network, which results in a steadier network behaviour.

The optimality of duty-cycle assignment in a given cluster-tree network typically depends on the optimized metric (e.g. bandwidth, buffer size, delay) and for which entries (e.g. traffic type, delay requirements). In this paper, we propose a general methodology for assigning a duty-cycle to each coordinator proportionally to its traffic needs to ensure a fair utilization of the bandwidth. This methodology can be easily adapted to any other metric by just considering the constraints related to that metric.

The idea of the efficient duty-cycle assignment consists in expressing the different network constraints of the duty-cycle for each coordinator and its relation to the duty-cycles of other coordinator. This results in a set of linear equations that can be easily resolved using the theory of linear algebra. The set of constraints depends on the configuration of the network. In what follows, we first consider the general case of unbalanced cluster-tree networks (Sect. 7.2) and then we provide a simpler result for balanced networks (Sect. 7.3). Note that in this section, we do not use the coordinator grouping technique and we assume that *the root coordinator* (ZC) *is the destination of all traffic*. Downstream traffic and data aggregation are not considered in our analysis.

7.2 Case of unbalanced cluster-tree WSNs

We propose to analyze the duty-cycle assignment for the worst-case situation in terms of network load, i.e. all nodes in the network generating the maximum amount of

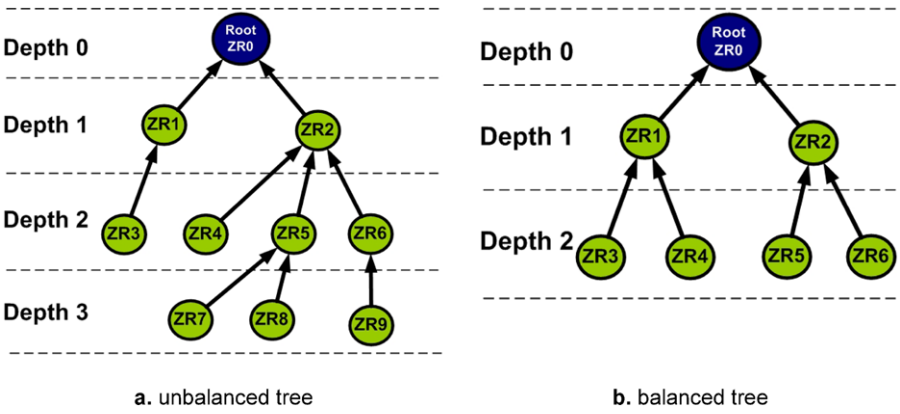


Fig. 8 Cluster-tree network examples

traffic simultaneously. Additionally, we assume that, in the worst-case, all ZigBee routers have *the same maximum number of child nodes* that *generate the same traffic shape* (e.g. using a leaky bucket shaping model). This is a realistic assumption since, in general, sensors of the same type generate nearly the same traffic and the efficient management of the duty-cycle is mainly necessary in the worst-case load situation.

In order to give some intuition on the duty-cycle assignment approach, let us consider the following cluster-tree WSN example in Fig. 8a. The cluster-tree network contains $N = 10$ ZigBee routers (including one ZC). We denote DC_i the duty-cycle of the ith router.

In order to adequately assign the duty-cycle to each coordinator, we must establish the set of equations that express the constraints on the duty-cycle of each router such that the duty-cycle is proportional to its incoming traffic. In the following, we derive three general conditions that must be satisfied by the duty-cycles.

- A *first constraint* is that the sum of all duty-cycles must be lower or equal to one. This sum must be equal to one if we aim to maximize the activity period of the cluster-tree network, or it can be set to any other value lower than one depending on the applications bandwidth requirements. Hence, we consider the following equation expressing this constraint:

$$\sum_{i=1}^{N} DC_i \leq 1. \tag{4}$$

- A *second constraint* is that the duty-cycle of a parent coordinator must be at least equal to the sum of all duty-cycles of its child coordinators. This is because the activity period of the parent coordinator must be able to support all the ingoing traffic from its children.

$$DC_{parent} - \sum DC_{children} \geq 0. \tag{5}$$

- A *third constraint* is that all leaf coordinators (the ones with no child coordinators) must have the same duty-cycle, even at different depths. This is because the

incoming traffic is the same in each leaf coordinator (note that we consider no data aggregation).

Now, applying these three conditions to the example in Fig. 8 we derive the required constraints for computing all the duty-cycles. For instance, applying (5) to the case of ZR0, ZR1 and ZR2, the duty-cycle DC_0 is equal to the sum of duty-cycles DC_1 and DC_2, since the ingoing traffic at the Root node (ZR0) is equal to the sum of the output traffic of routers ZR1 and ZR2. Hence, $DC_0 - DC_1 - DC_2 = 0$. Applying the same principle to all ZigBee routers in the cluster-tree network we obtain a system of linear equations with $N = 10$ unknown variables (duty-cycles) that can be resolved using the linear algebra theory. Since we can have at least one independent equation for each router, the number of equations is at least equal to the number of unknown variables, thus the linear equation system has only one solution, if it exists. The linear equation system can be easily re-written in a matrix form as $A \cdot DC^T = b$ where $DC = [DC_0, DC_1, \ldots, DC_{N-1}]$ is the duty-cycle vector of all ZigBee routers. Note that DC^T is the transpose of DC.

The matrix is square if the number of equations is equal to the number of unknown variables. The solution of such a linear system is then:

$$DC = A^{-1} \cdot b, \quad \text{where } A^{-1} = \frac{A^T}{\det(A)} \text{ for } \det(A) \neq 0. \tag{6}$$

It is also possible to resolve the linear equation system even if the system is not square. Computational tools such as MATLAB easily process such equation system.

Applying this methodology for the above example, we get the following square equation system.

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} DC_0 \\ DC_1 \\ DC_2 \\ DC_3 \\ DC_4 \\ DC_5 \\ DC_6 \\ DC_7 \\ DC_8 \\ DC_9 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \tag{7}$$

Observe that the first line in the Matrix corresponds to (4), the lines from 2 to 6 correspond to the second constraint in (5) and the remaining lines correspond to the third constraint. The solution to the linear equation system is then:

$$\begin{aligned} DC &= \begin{bmatrix} DC_0, DC_1, DC_2, DC_3, DC_4, \\ DC_5, DC_6, DC_7, DC_8, DC_9 \end{bmatrix} \\ &= \begin{bmatrix} 0.2777, 0.0555, 0.2222, 0.0555, 0.0555, \\ 0.1111, 0.0555, 0.0555, 0.0555, 0.0555 \end{bmatrix}. \end{aligned} \tag{8}$$

Note that according to (1) and (2), the duty-cycle must be a power of 2. Hence, the valid values of the duty-cycle are:

$$\overline{DC} = 2^{IO}, \quad \text{where } IO = \lfloor \log_2(DC) \rfloor. \tag{9}$$

It follows that:

$$\overline{DC} = \begin{bmatrix} 2^{-2}, 2^{-5}, 2^{-3}, 2^{-5}, 2^{-5} \\ 2^{-4}, 2^{-5}, 2^{-5}, 2^{-5}, 2^{-5} \end{bmatrix}. \tag{10}$$

Hence, we accurately determined the duty-cycles that ensure the fairest bandwidth distribution among all the routers of the cluster-tree network. It is possible to choose a fixed beacon order for the whole network and vary the superframe order of each router to fit the corresponding duty-cycle in (10).

7.3 Case of balanced cluster-tree WSNs

Observe that this methodology applied to the case of unbalanced trees is quite efficient to determine exactly the required duty-cycle for each coordinator, according to the traffic that crosses it in a particular tree configuration. Nonetheless, when a new coordinator joins the tree, the duty-cycle assignment must be re-computed again, which induces a limited flexibility to dynamic network changes. This limitation can be overcome by dimensioning the duty-cycle assignment for the worst-case cluster-tree network configuration, which has the maximum depth and the maximum number of ZigBee routers.

According to the ZigBee standard, each parent coordinator has a maximum number of child coordinators that can be associated to it. Assuming that all coordinators are equivalent, this leads to a symmetric (or balanced) cluster-tree network with a maximum depth *maxDepth* and a maximum number of child routers associated to a parent router, N_{router}. The example in Fig. 8b presents a balanced tree with $maxDepth = 2$ and $N_{router} = 2$. The leaf routers are those at the lowest depth. This worst-case network model has been adopted in Koubaa et al. (2006). It comes that the motivation of considering this model is two-folded, as described next.

First, the duty-cycle assignment is computed only once for this worst-case network configuration and each time a new coordinator joins the network, in any configuration, it will be assigned the duty-cycle corresponding to the worst-case cluster-tree network. Thus, there is no need to generate a new schedule with the SDS algorithm when a new coordinator joins the network (oppositely to the unbalanced tree case).

Second, the duty-cycle dimensioning is much simpler than for the unbalanced tree case, since the duty-cycle will only depend on the depth of the coordinators, i.e. all coordinators at the same depth have the same duty-cycle.

In case of balanced cluster-tree networks, the problem is then reduced to finding the duty-cycles for each depth. It results that the duty-cycle of each router is equal to the duty-cycle for a given depth divided by the number of routers at that depth. Let us denote DC_i the duty-cycle of a router at depth i. Observe that the duty-cycle of a router at depth i can be expressed as a function of DC_0 as:

$$DC_0 = N_{router}^{i} \cdot DC_i. \tag{11}$$

Based on (4) and (9), we obtain:

$$DC_0 = 2^{\lfloor \log_2(\frac{1}{maxDepth}) \rfloor}. \quad (12)$$

Equations (11) and (12) enable the computation of all the duty-cycles in the cluster-tree network.

For any network configuration where the depth is smaller than *maxDepth* and the number of child routers per parent router is smaller than N_{router}, the duty-cycle assignment will not change, which significantly improves the flexibility to dynamic network changes.

7.4 Discussion on re-synchronization issues in cluster-tree WSNs

The run-time reconfiguration of cluster-tree networks is an important requirement in order to cope with its dynamics (e.g. node/cluster mobility/relocation, physical/logical, increasing/decreasing cluster duty-cycles,). In a cluster-tree network, this can imply the dynamic duty-cycle assignment according to the bandwidth required by each cluster at a given time, along with the reorganization of beacon transmissions (new beacon/superframe scheduling).

The TDBS mechanism solves the problem of beacon/superframe admission control and scheduling, according to the bandwidth requirements of each cluster. While applying the resulting beacon/superframe schedule to the network pre-run-time is a trivial task, in dynamic networks where nodes/clusters can join/leave the network during run-time (e.g. mobility, battery depletion) there is the need to tackle the problem or network re-synchronization, i.e. applying the new schedule to every cluster in the network.

We will pursue two network re-synchronization approaches: one is to centralize the cluster-tree scheduling in the ZigBee Coordinator (the root of the cluster-tree) leading to the full re-scheduling of the tree;, another is to distribute the scheduling among the ZigBee Routers (cluster heads) and only re-schedule a part of the tree. Nevertheless, the initial (pre-run-time) scheduling and synchronization of the cluster-tree is needed before triggering any re-synchronization mechanism during run-time. This is a challenging task as, in both cases, it is necessary to define appropriate mechanisms to handle the re-synchronization, such as: the admission control mechanism (either distributed in each ZigBee Router or centralized in the ZigBee Coordinator) and the re-synchronization mechanism to trigger and deploy the new scheduling. Importantly, re-synchronization must be accomplished in a way that minimizes network inaccessibility times and energy consumption.

A centralized re-synchronization approach can assume that the root (ZigBee Coordinator) has full knowledge of the network topology along with the information about all routers in the network. Thus, each node joining or leaving the network must inform the root, that in turn generates and dispatches a new schedule (based on the received requests) by triggering the re-synchronization procedure.

The distributed re-synchronization approach can assume that all routers are responsible for the re-synchronization task of their child routers. Consequently, this

allows that each router can manage its assigned transmission space (or time windows) where their child routers are allocated. This local scheduling enables the re-synchronization of only a part of the network along with a local management of the time assigned to each router. This enables the root to dynamically adapt different schedules by just allowing each cluster to shrink/grow its active period within a maximum predefined window.

8 Experimental evaluation

The purpose of this section is to demonstrate the feasibility of the time division scheduling mechanism described in Sect. 6, using our own implementation of the IEEE 802.15.4/ZigBee protocol stack (Cunha et al. 2007b; Open-zb 2007) in nesC (Gay et al. 2003) for TinyOS (v1.15) (TinyOS 2007) supporting the TelosB (Crossbow 2007) motes.

8.1 The open-ZB IEEE 802.15.4/ZigBee protocol stack

The open-ZB protocol stack (Cunha et al. 2007b; Open-zb 2007) has been used to leverage our research work, in the sense that it has been enabling to test, validate and demonstrate our scientific findings through experimentation, including the time division beacon scheduling mechanism presented in Sect. 6. This work has also been driven by the need for an open-source implementation of the IEEE 802.15.4/ZigBee protocols, filling a gap between some newly released complex *C* implementations and black-box implementations from different manufacturers. Currently, the open-ZB implements the beacon-enabled mode of the IEEE 802.15.4 protocol and the ZigBee Network Layer core functionalities, supporting the Crossbow MICAz and TelosB motes (Crossbow 2007). Importantly, this open-source implementation has been and will continue to potentiate research works on the IEEE 802.15.4/ZigBee protocols and WSNs in general, allowing their demonstration and validation through experimentation. In this context, the open-ZB website (Open-zb 2007) offers an open-source toolset for these protocols, namely the referred protocol stack implementation and an OPNET (http://www.opnet.com) simulation model (Jurčík and Koubâa 2007) of the IEEE 802.15.4. In the medium term, it is envisaged to implement the full IEEE 802.15.4 protocol stack and the full functionalities of the ZigBee Network Layer.

We are currently addressing the migration of the ZigBee Cluster-Tree implementation from TinyOS v1.15 to TinyOS v2.0, as a result from our collaboration with the TinyOS Network Protocol Working Group (http://tinyos.stanford.edu:8000/Net2WG) to implement a ZigBee compliant stack for TinyOS 2.0.

8.2 TDBS implementation approach

The Time Division Beacon Scheduling (TDBS) mechanism can be implemented in a simple manner, with only minor add-ons to the protocol. For a more detailed description of the implementation refer to Cunha et al. (2007a).

The implementation of this mechanism assumes the following:

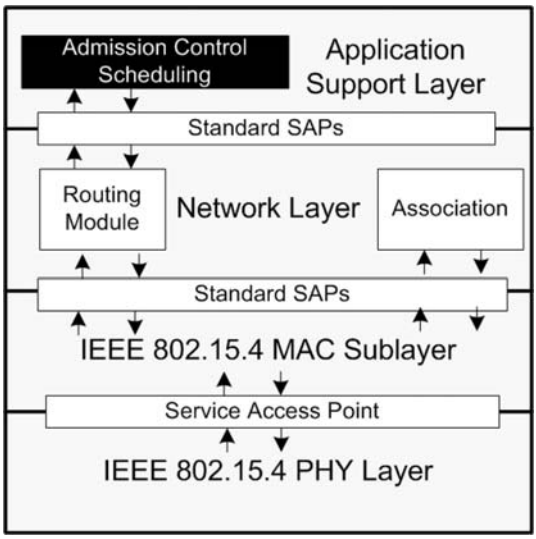


Fig. 9 Time division beacon scheduling implementation architecture

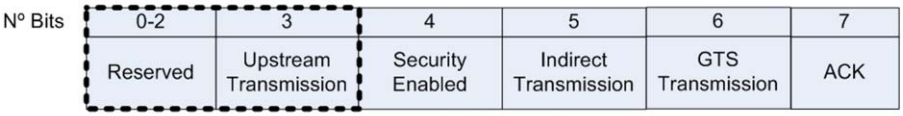
1. The ZigBee Network Layer supports the tree-routing mechanism, thus the network addresses of the devices are assigned accordingly.
2. The ZigBee Coordinator is the first node broadcasting beacons in the network.
3. The ZigBee Routers start to send beacons only after a successful negotiation.
4. The same Beacon Interval (BI) is used by every ZigBee Router. Note that this just a simple choice to simply the implementation, without loss of generality. This would also be feasible for the case of different *BI*s, but with a slightly higher implementation complexity.

The TDBS approach relies on a negotiation prior to beacon transmission. Upon success of the association to the network, a ZR (initially behaving as a ZED) sends a negotiation message to the ZC (routed along the tree) embedding the envisaged (*BO*, *SO*) pair, requesting a beacon broadcast permit. Then, in the case of a successfully negotiation, the ZC replies with a negotiation response message containing a beacon transmission offset (the instant when the ZR must start transmitting the beacon). In case of rejection, the ZR must disassociate from the network.

Figure 9 depicts the architecture of the TDBS implementation in the IEEE 802.15.4/ZigBee protocol stack. The Admission Control and Scheduling algorithm fits as a service module of the Application Support Layer. The TDBS requires minor changes to the Network Layer. It is necessary to add a *StartTime* argument to the *MLME-START.request* primitive, as already proposed in the ZigBee Specification, and to the *NLME-START-ROUTER.request* primitive. The *StartTime* parameter will be used as a transmission offset referring to the parent ZigBee Router (ZR). In the ZC, the value of this parameter is 0.

After a successful negotiation of the beacon transmission, the ZR will have two active periods: its own (the superframe duration) and its parent's superframe duration. In its own active period, the ZR is allowed to transmit frames to its child nodes or relay frames to the descendant devices in the tree. The frames destined upstream are sent during its parent's active period. For this to be possible, there is the need to

a. IEEE 802.15.4 transmit options field



b. Time Division Beacon Scheduling transmit options field

Fig. 10 Transmit options field comparison

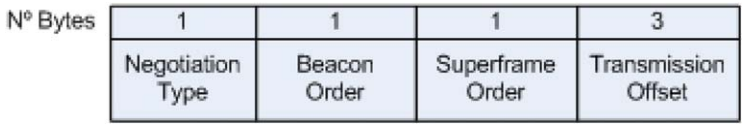


Fig. 11 Time division beacon scheduling negotiation field

implement a different buffer mechanism for each message flow - the downstream to the ZR descendants and the upstream to the ZR ascendants.

The buffer mechanism is implemented in the MAC sub-layer, using the downstream buffer or the upstream buffer depending of the transmission options parameter of the *MCPS_DATA.request* primitive. The transmit options or *TxOptions* parameter, last argument of the primitive defines the transmission options for the data frame, allowing the frame to be sent either in the CFP (GTS) or in the CAP (Slotted CSMA/CA). This parameter also defines if the transmission uses the upstream or the downstream buffer, as depicted in Fig. 10.

During the ZR superframe, all the messages that need to be transmitted to its parent are stored in the upstream buffer. During its parent's superframe, the ZR tries to transmit the messages upstream.

The ZR must therefore wake up during its parent's superframe. Two new timer events must be added to the MAC sub-layer. One is triggered at the beginning of the parent's superframe and turns on the transceiver in receive mode and another at the end turning the transceiver off.

Guaranteeing the synchronization of all network nodes is a big challenge, since all nodes must always be synchronized with their parents.

The negotiation of the beacon transmission is performed through a simple protocol that uses the data frames payload with a predefined format.

Figure 11 depicts the Time Division Beacon Scheduling negotiation frame format, which includes the following fields:

- *Negotiation type*—Indicates the type of the negotiation command. This field can have the following values: 1 for a negotiation request, 2 for a negotiation accept and 3 for a negotiation deny;
- *Beacon Order*—Indicates the beacon order of the ZR device;

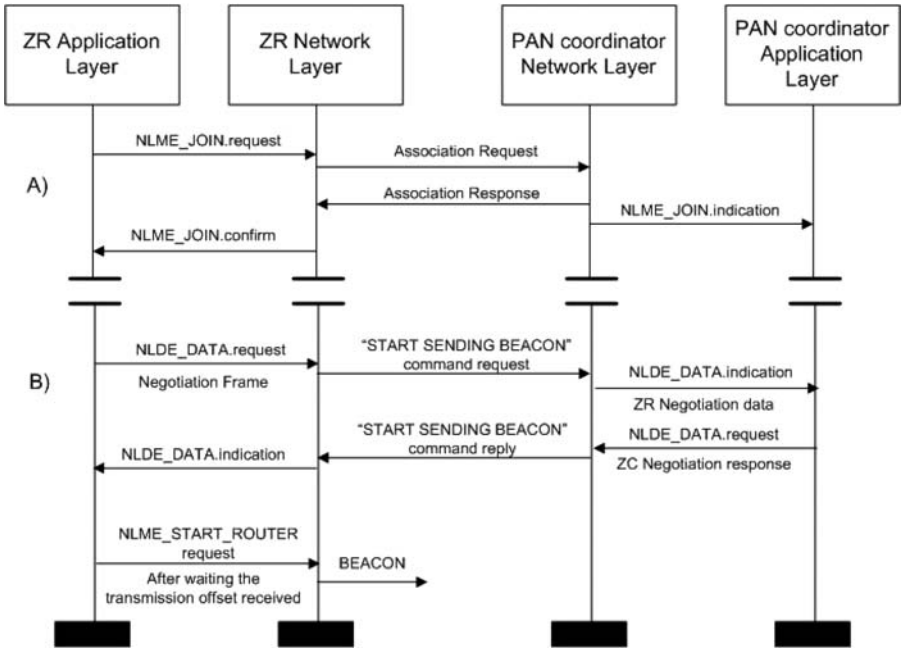


Fig. 12 Time division beacon scheduling negotiation diagram

- *Superframe Order*—Indicates the superframe order of the ZR device;
- *Transmission Offset*—Indicates the transmission offset defined by the ZC in a negotiation accept command.

In the negotiation request, the Beacon Order and Superframe Order fields indicate the ZR superframe configuration that is intended by the ZR candidate. In the negotiation response, the configuration may not be the same as the one requested. Instead, the ZC can assign a different *BO*, *SO* configuration according to its available resources.

Figure 12 depicts a diagram with the sequence of Network Layer events from the association of the ZR (Part A) until the beacon transmission after a successful negotiation (Part B).

8.3 Network scenario

This section presents the results obtained from the implementation of the time division beacon scheduling approach. This experimental work demonstrates the feasibility of this approach using TelosB motes. Other experiments are presented in Cunha et al. (2007a).

The cluster-tree network scenario used in the following experiments is presented in Fig. 13.

The network contains one ZC and 14 ZRs (15 clusters total). The Beacon Order (*BO*) is set to 8 for all Coordinators, which corresponds to a Beacon Interval

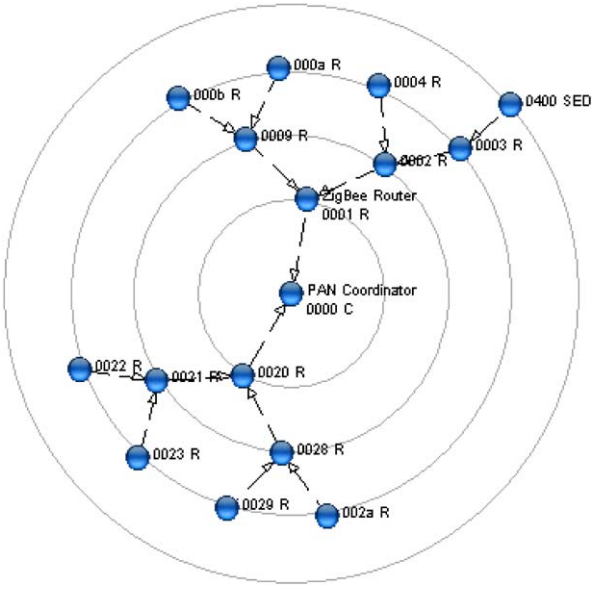


Fig. 13 Experimental cluster-tree network configuration

of 245760 symbols (≅ 4122.624 ms). Hence, we must have at least $2^4 = 16$ Beacon/Superframe time windows, each with duration of 15360 symbols (≅ 257.664 ms). This restricts the (maximum) Superframe Order (*SO*) to 4 (i.e. Superframe Duration (*SD*) = 15360 symbols (≅ 257.664 ms)), which was actually our option for these experiments. The ZigBee cluster-tree network parameters (for setting up the tree routing mechanism) consist in a maximum depth (*Lm*) equal to $Lm = 3$, a maximum number of child nodes per parent router (*Cm*) equal to $Cm = 6$, and a maximum number of child routers per parent router (*Rm*) equal to $Rm = 4$. As shown in Fig. 13, the network comprises the ZC at depth 0, two ZRs at depth 1, four ZRs at Depth 2 and eight ZRs at depth 3. One ZED (0x0400) was also considered for performing a message routing test.

The network topology presented in Fig. 13 was automatically generated by the Daintree IEEE 802.15.4/ZigBee Network/Protocol Analyser application (Daintree Networks 2006), upon the association of all ZRs and the ZED with their respective parents.

Note that in our implementation the duration of two symbols (8 bits) approximately corresponds to 33.55 μs, which is slightly different from the theoretical value of 32 μs specified by the IEEE 802.15.4 standard (IEEE-TG15.4 2003). This inaccuracy resulting from the hardware timer constraint is due to the internal timer clock tick granularity of the TelosB mote and leads to a cumulative effect on the discrepancy between the theoretical and experimental values of the beacon interval. The requirement that best fits this implementation is to trigger the timer on every backoff. The IEEE 802.15.4 standard defines one backoff as 20 symbols, that theoretically correspond to 16 μs. With the TelosB clock granularity, the value obtained for each symbol is approximately 16.775 μs, leading to a backoff period duration of 335.5 μs instead of the 320 μs defined in the IEEE 802.15.4 standard. For instance, the beacon interval (BI) for $BI = 8$ is equal to 245760 symbols, which theoretically corresponds

Table 3 Beacon schedule (for the 15 clusters)

Time Window	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Short Address (0x00..)	00	01	02	03	04	09	0a	0b	20	21	22	23	28	29	2a	–

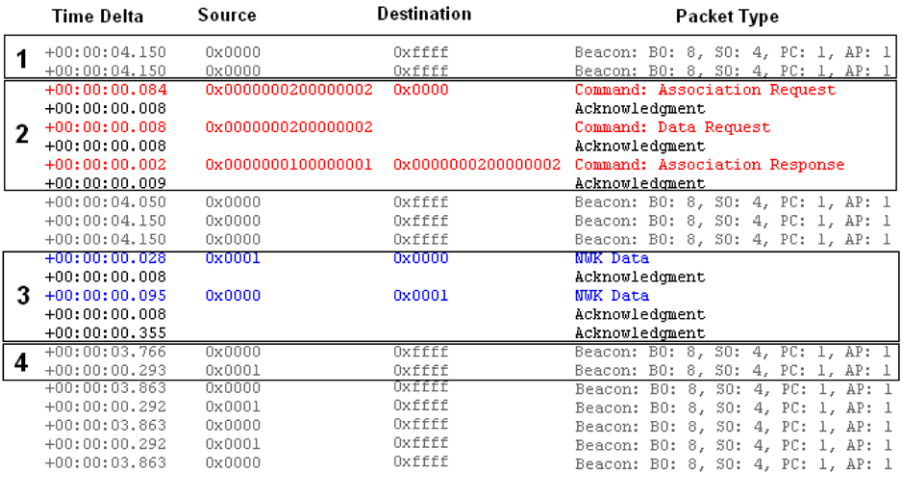


Fig. 14 Association and negotiation example

to 3932.160 ms, but experimentally corresponds to 4122.624 ms. This discrepancy, however, does not impact the correct behaviour of the implemented protocol.

8.4 Beacon scheduling and network formation

Table 3 shows the schedule for the beacon interval. Each associated ZR will be assigned the respective time window according to their position in the network tree. For viewing purposes, only the last 2 bytes of the short addresses are shown.

Figure 14 presents a frame sequence chart grabbed using a Chipcon sniffer (Chipcon Packet Sniffer 2006). The beacon frame sent by the ZC (marked as 1), contains the network configuration (*BO*, *SO*), as seen in the *Packet Type* field. Note that the Time Delta (4150 ms) between beacons represents the beacon interval. The sequence of messages marked as 2 represents the association procedure. The ZR with the extended address of 0x0000000200000002 sends an association request to the ZC (0x0000). The ZC acknowledges the reception of the request and informs the ZR that there is pending data (using the pending data field in the acknowledge frame). Then, the ZR sends a data request command frame requesting the pending data. The ZC replies with the association response command frame containing the status of the association (that in this case is successful) and the ZR is assigned the short address 0x0001.

Now, the ZR is associated as a ZED and can therefore communicate normally, but it still needs to request the ZC for a beacon broadcast transmission permit and

Frame 15 (Length = 27 bytes)	Frame 17 (Length = 27 bytes)
Time Stamp: 14:53:34.863	Time Stamp: 14:53:34.867
Frame Length: 27 bytes	Frame Length: 27 bytes
Capture Length: 27 bytes	Capture Length: 27 bytes
Link Quality Indication: 136	Link Quality Indication: 164
Receive Power: -49 dBm	Receive Power: -42 dBm
IEEE 802.15.4	IEEE 802.15.4
Frame Control: 0x8821	Frame Control: 0x8821
Sequence Number: 165	Sequence Number: 34
Destination PAN Identifier: 0x1234	Destination PAN Identifier: 0x1234
Destination Address: 0x0000	Destination Address: 0x0001
Source PAN Identifier: 0x1234	Source PAN Identifier: 0x1234
Source Address: 0x0001	Source Address: 0x0000
Frame Check Sequence: Correct	Frame Check Sequence: Correct
ZigBee NWK	ZigBee NWK
Frame Control: 0x0004	Frame Control: 0x0004
Destination Address: 0x0000	Destination Address: 0x0001
Source Address: 0x0001	Source Address: 0x0000
Radius = 1	Radius = 1
Sequence Number = 97	Sequence Number = 79
NWK Payload: 01:08:04:00:00:00	NWK Payload: 02:08:04:00:3c:00
a) Negotiation request	**b) Negotiation response**

Fig. 15 Negotiation mechanism packet decode example

a time window slot (transmission offset). The negotiation procedure is marked as 3. Until this point, and after the network association, the ZR behaves as a normal ZED. When the negotiation for beacon transmission finishes (successfully), the ZR starts to broadcast beacons in its assigned time window, as seen in Fig. 14 marked as 4. Note that both the association and negotiation for beacon transmission takes place during the ZC superframe.

Figure 15 shows the negotiation packets in a decoded format. In Fig. 15a the negotiation request (from ZR 0x0001 to ZC 0x0000) and in Fig. 15b we can find the negotiation accept (from ZC 0x0000 to ZR 0x0001). Highlighted is the data frame payload, in green (the first byte) is the negotiation type of message, in blue (the second and third bytes) the information of the beacon and superframe orders and in yellow (fourth to sixth bytes) the beacon transmission offset value in symbols.

The ZigBee End Device 0x0400 has associated with ZR 0x0003 with the purpose of periodically transmitting data frames through the cluster-tree, in order to test the topology and the tree-routing mechanism. In Fig. 16, the message flows are marked with capital characters (e.g. A, B, C) and the hop count with numbers (e.g. A1, A2, A3).

In Fig. 16, marked as A1, the first transmission of the packet from the ZED (0x0400) to its parent (ZR 0x0003) is shown. Note that this transmission is carried

out during ZR 0x0003 superframe. The routing of the data frame from ZR 0x0003 to its parent in the cluster-tree (ZR 0x0002) is marked as A2. The multi-hop continues with the routing of the frame from ZR 0x0002 to ZR 0x0001 (A3). In B1, a new message flow is initiated by the ZED (0x0400). Then, in A4, the message is relayed from ZR 0x0001 to ZC (0x0000) and to ZR 0x0020. This transmission sequence is carried out during the ZC superframe. The multi-hop continues in A5 between ZR 0x0020 and ZR 0x0028. The last hop is carried out in A6 with ZR 0x0028 relaying it to its final destination, ZR 0x0028.

9 Concluding remarks

While the IEEE 802.15.4/ZigBee protocol specifications support both mesh and cluster-tree multiple cluster topologies, only the former has been completely defined and is commercially available. Basically, the mesh networking solution is attractive because it provides a flexible, redundant and simple engineering solution to multiple cluster sensor networks. The ZigBee cluster-tree solution is more difficult to engineer, since it requires a distributed synchronization scheme that is prone to the single point of failure problem in the ZigBee Routers (cluster-heads). However, ZigBee cluster-tree networks allow the per-cluster dynamic allocation of guaranteed bandwidth and duty-cycles, enabling real-time and energy-efficient communications in wireless sensor networks.

Despite these appealing features, ZigBee cluster-tree topologies have been completely neglected by the ZigBee Alliance and related companies, which have been exclusively investing in mesh topologies so far. The reasons for this commercial strategy are, to our best knowledge: 1) ZigBee mesh topologies are much easier to engineer—they scale more easily, enable path redundancy, do not implement synchronization; 2) the cluster-tree network model is still open in the IEEE 802.15.4/ZigBee protocol specifications; and 3) while the mesh topology is not energy-efficient (does not allow router nodes to sleep), current major ZigBee Alliance target applications (domotics and electrical energy telemetry) are not so dependent on batteries, since it is possible to connect at least a subset of the sensor nodes to the electrical grid, thus not compromising network lifetime.

Taking into account that it is widely accepted that future distributed embedded applications will increasingly require time and energy efficiency, such as in cyber-physical systems, we have been betting on filling the gaps in the ZigBee cluster-tree network specification. In this paper, we provided a solution to a real fundamental problem in the IEEE 802.15.4/ZigBee protocols: how to engineer a cluster-tree network. We proposed the Time Division Beacon/superframe Scheduling (TDBS) algorithm, which efficiently assigns distinct time windows to adjacent clusters, in order to avoid inter cluster (beacon, data) frame collisions. The feasibility of the TDBS algorithm has been demonstrated via an experimental test-bed involving a cluster-tree network with 15 clusters. In addition, an efficient duty-cycle management mechanism for ensuring a fair distribution of bandwidth resources has been presented. This mechanism is quite useful for an optimized dimensioning of network resources for achieving better performance.

	Time	Time Delta	Source	Destination	NWK Source	NWK Destination	Protocol	Packet Details
	16:53:43.940	+00:00:00.287	0x002a	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:44.136	+00:00:00.196	0x0000	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:44.428	+00:00:00.292	0x0001	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:44.720	+00:00:00.292	0x0002	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
A 1	16:53:44.977	+00:00:00.256	0x0400	0x0003	0x0400	0x0029	Zigbee NWK	NWK Data
	16:53:44.985	+00:00:00.009					IEEE 802.15.4	Acknowledgment
	16:53:45.076	+00:00:00.090	0x0003	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:45.311	+00:00:00.235	0x0004	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:45.594	+00:00:00.283	0x0009	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:45.886	+00:00:00.292	0x000a	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:46.173	+00:00:00.287	0x000b	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:46.450	+00:00:00.277	0x0020	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:46.742	+00:00:00.292	0x0021	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:47.034	+00:00:00.293	0x0022	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:47.321	+00:00:00.287	0x0023	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:47.603	+00:00:00.282	0x0028	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:47.895	+00:00:00.292	0x0029	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:48.183	+00:00:00.287	0x002a	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:48.378	+00:00:00.196	0x0000	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:48.670	+00:00:00.292	0x0001	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:48.963	+00:00:00.292	0x0002	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
A 2	16:53:48.975	+00:00:00.013	0x0003	0x0002	0x0400	0x0029	Zigbee NWK	NWK Data
	16:53:48.984	+00:00:00.008					IEEE 802.15.4	Acknowledgment
	16:53:49.261	+00:00:00.277	0x0003	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:49.554	+00:00:00.293	0x0004	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:49.836	+00:00:00.282	0x0009	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:50.128	+00:00:00.292	0x000a	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:50.415	+00:00:00.287	0x000b	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:50.692	+00:00:00.277	0x0020	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:50.984	+00:00:00.292	0x0021	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:51.276	+00:00:00.292	0x0022	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:51.563	+00:00:00.287	0x0023	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:51.845	+00:00:00.282	0x0028	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:52.137	+00:00:00.292	0x0029	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:52.425	+00:00:00.287	0x002a	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:52.620	+00:00:00.195	0x0000	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:52.912	+00:00:00.292	0x0001	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
A 3	16:53:52.925	+00:00:00.013	0x0002	0x0001	0x0400	0x0029	Zigbee NWK	NWK Data
	16:53:52.933	+00:00:00.008					IEEE 802.15.4	Acknowledgment
	16:53:53.211	+00:00:00.277	0x0002	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
B 1	16:53:53.483	+00:00:00.273	0x0400	0x0003	0x0400	0x0029	Zigbee NWK	NWK Data
	16:53:53.492	+00:00:00.009					IEEE 802.15.4	Acknowledgment
	16:53:53.566	+00:00:00.075	0x0003	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:53.802	+00:00:00.235	0x0004	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:54.089	+00:00:00.287	0x0009	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:54.381	+00:00:00.292	0x000a	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:54.668	+00:00:00.287	0x000b	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:54.945	+00:00:00.277	0x0020	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:55.238	+00:00:00.292	0x0021	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:55.530	+00:00:00.292	0x0022	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:55.817	+00:00:00.287	0x0023	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:56.099	+00:00:00.282	0x0028	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:56.391	+00:00:00.292	0x0029	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:56.678	+00:00:00.287	0x002a	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:56.874	+00:00:00.196	0x0000	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:56.887	+00:00:00.013	0x0001	0x0000	0x0400	0x0029	Zigbee NWK	NWK Data
A 4	16:53:56.895	+00:00:00.008					IEEE 802.15.4	Acknowledgment
	16:53:56.929	+00:00:00.034	0x0000	0x0020	0x0400	0x0029	Zigbee NWK	NWK Data
	16:53:56.937	+00:00:00.008					IEEE 802.15.4	Acknowledgment
	16:53:57.184	+00:00:00.247	0x0001	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:57.476	+00:00:00.292	0x0002	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
B 2	16:53:57.489	+00:00:00.013	0x0003	0x0002	0x0400	0x0029	Zigbee NWK	NWK Data
	16:53:57.498	+00:00:00.009					IEEE 802.15.4	Acknowledgment
	16:53:57.775	+00:00:00.277	0x0003	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:58.067	+00:00:00.293	0x0004	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:58.350	+00:00:00.282	0x0009	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:58.642	+00:00:00.292	0x000a	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:58.929	+00:00:00.287	0x000b	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:59.275	+00:00:00.347	0x0020	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
A 5	16:53:59.278	+00:00:00.002	0x0020	0x0028	0x0400	0x0029	Zigbee NWK	NWK Data
	16:53:59.292	+00:00:00.014					IEEE 802.15.4	Acknowledgment
	16:53:59.579	+00:00:00.287	0x0021	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:53:59.871	+00:00:00.292	0x0022	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:54:00.158	+00:00:00.287	0x0023	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:54:00.504	+00:00:00.346	0x0028	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
A 6	16:54:00.506	+00:00:00.002	0x0028	0x0029	0x0400	0x0029	Zigbee NWK	NWK Data
	16:54:00.520	+00:00:00.014					IEEE 802.15.4	Acknowledgment
	16:54:00.829	+00:00:00.309	0x0029	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:54:01.095	+00:00:00.265	0x002a	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:54:01.183	+00:00:00.089	0x0000	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:54:01.476	+00:00:00.292	0x0001	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
B3	16:54:01.488	+00:00:00.013	0x0002	0x0001	0x0400	0x0029	Zigbee NWK	NWK Data
	16:54:01.497	+00:00:00.008					IEEE 802.15.4	Acknowledgment
	16:54:01.774	+00:00:00.277	0x0002	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
C 1	16:54:01.975	+00:00:00.202	0x0400	0x0003	0x0400	0x0029	Zigbee NWK	NWK Data
	16:54:01.984	+00:00:00.008					IEEE 802.15.4	Acknowledgment
	16:54:02.129	+00:00:00.145	0x0003	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:54:02.365	+00:00:00.236	0x0004	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1
	16:54:02.652	+00:00:00.288	0x0009	0xffff			IEEE 802.15.4	Beacon: BO: 8, SO: 4, PC: 1, AP: 1

Fig. 16 Message flow—data frame multi-hop thought the cluster-tree

We believe that this work represents a milestone towards the use of the ZigBee cluster-tree topology in wireless sensor/actuator networks with energy and timing requirements, filing an existing gap in the IEEE 802.15.4/ZigBee standards. To our best knowledge, this was not solved in previous works, neither from academia nor from industry.

We have recently experimentally evaluated the performance of ZigBee cluster-tree networks against the theoretical worst-case results obtained in Koubaa et al. (2006), supporting sinks with mobile behaviour (Jurčík et al. 2008) and overcoming some of the inherent limitations of the technological platforms (Cunha et al. 2008). We now intend to find a simple yet effective mechanism for applying new schedules (re-synchronization) to all clusters with minor energy consumption and network inaccessibility times, upon dynamic network changes (nodes/clusters moving or joining/leaving the network). As a longer term objective, we envisage to investigate mechanisms for overcoming the single point of failure problem in cluster-tree topologies.

References

Adams J (2005) Building low power into wireless sensor networks using ZigBee technology. In: Industrial embedded systems resource guide, networking: technology, pp 26–30

Caccamo M, Zhang LY (2002) An implicit prioritized access protocol for wireless sensor networks. In: Proceedings of the 23rd IEEE real-time systems symposium (RTSS'02), Austin, Texas, 2002

Chipcon Packet Sniffer for IEEE 802.15.4 v1.0 (2006)

Crossbow (2007) http://www.xbow.com

Culter T (2005) Deploying ZigBee in existing industrial automation networks. In: Industrial embedded system resource guide, networking: technology, pp 34–36

Cunha A (2007) On the use of IEEE 802.15.4/ZigBee as federating communication protocols for wireless sensor networks. Master thesis, July. Available at http://www.hurray.isep.ipp.pt/ART-WiSe

Cunha A, Alves M, Koubaa A (2007a) Implementation details of the time division beacon frame scheduling approach for zigbee cluster-tree networks, IPP-HURRAY Technical Report TR070102. http://www.open-zb.net

Cunha A, Koubâa A, Severino R, Alves M (2007b) Open-ZB: an open-source implementation of the IEEE 802.15.4/ZigBee protocol stack on TinyOS. In: Proc of the 4th IEEE international conference on mobile ad-hoc and sensor systems (MASS'07), Pisa, Italy, October 2007

Cunha A, Severino R, Pereira N, Koubâa A, Alves M (2008) ZigBee over TinyOS: implementation and experimental challenges, to be published at the 8th Portuguese conference on automatic control—CONTROLO'2008, Invited session on "real-time communications: from theory to applications, Vila Real, Portugal, July 2008

Daintree Networks (2006) Sensor network analyser. http://www.daintree.net

Diestel R (2005) Graph theory, 3rd edn. Springer, New York

Gay D, Levis P, von Behren R, Welsh M, Brewer E, Culler D (2003) The nesC language: a holistic approach to networked embedded systems. In: Programming language design and implementation

Geer D (2005) Users make a beeline for ZigBee technology. IEEE Comput 38(12):16–19

Gupta G, Younis M (2003) Fault-tolerant clustering of wireless sensor networks. In: IEEE wireless communication and networks conference (WCNC 2003), vol 3. New Orleans, Louisiana, 2003

Ha J, Kwon WH, Kim JJ, Kim YH, Shin YH (2005) Feasibility analysis and implementation of the IEEE 802.15.4 multi-hop beacon-enabled network. In: 5th joint conf on communications & info, 2005

Heinzelman WR, Chandrakasan A, Balakrishnan H (2000) Energy-efficient communication protocols for wireless microsensor networks. In: Proceedings of the Hawaii international conference on systems sciences, Hawai, 2000

Herzog C (2005) Creating value with ZigBee networks. In: Industrial embedded system resource guide, networking: technology, pp 31–33

Holte R, Mok A, Rosier L, Tulchinsky I, Varvel D (1989) The pinwheel: a real-time scheduling problem. In: Proceedings of the 22nd Hawaii international conference on system science, Hawaii, 1989

IEEE-TG15.4 (2003) Part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs). IEEE standard for information technology

IEEE802.15.4b (2006) Draft revision for IEEE standard for information technology—telecommunications and information exchange between systems—local and metropolitan area networks—specific requirements—Part 15.4b: wireless medium access (MAC) and physical layer (PHY) specifications for low rate wireless personal area networks (WPANs). To be publically available in September 2006

Jurčík P, Koubâa A (2007) The IEEE 802.15.4 OPNET simulation model: reference guide v2.0. www.open-zb.net. IPP-HURRAY Technical Report, HURRAY-TR-070509, May 2007

Jurčík P, Severino R, Koubâa A, Alves M, Tovar E (2008) Real-time communications over cluster-tree sensor networks with mobile sink behaviour. IPP-HURRAY Technical Report, HURRAY-TR-080401, May. Submitted to a conference

Kottapalli VA, Kiremidjiana AS, Lyncha JP, Carryerb E, Kennyb TW, Law KH, Lei Y (2003) Two-tiered wireless sensor network architecture for structural monitoring. In: Proceedings of the 10th annual international symposium on smart structures and materials, San Diego (USA), 2003

Koubaa A, Alves M, Tovar E (2006) Modeling and worst-case dimensioning of cluster-tree wireless sensor networks. In: 27th IEEE real-time systems symposium (RTSS'06), Rio de Janeiro, Brazil, 2006

Mirza D, Owrang M, Schurgers C (2005) Energy-efficient wakeup scheduling for maximizing lifetime of IEEE 802.15.4 networks. In: First international conference on wireless internet (WICON'05), Budapest (Hungary), 2005

Open-zb, an open source toolset for the IEEE 802.15.4/ZigBee protocol stack (2007) http://www.open-zb.net

OPNET Technologies Inc, Opnet modeler wireless suite—ver 11.5A. http://www.opnet.com

Rajendran V, Obraczka K, Garcia-Luna-Aceves JJ (2003) Energy-efficient, collision-free medium access control for wireless sensor networks. In: Proceedings of the 1st international conference on embedded networked sensor systems (SenSys '03), California (USA), 2003

Rowe A, Mangharam R, Rajkumar R (2006) Rt-link: A time synchronized link protocol for energy constrained multi-hop wireless networks. In: Third annual IEEE communications society conference on sensor, mesh and ad hoc communications and networks (SECON), USA, 2006

TinyOS (2007) http://www.tinyos.net

Zheng J, Myung JL (2004) Will IEEE 802.15.4 make ubiquitous networking a reality?—A discussion on a potential low power, low bit rate standard. IEEE Commun Mag 42(6):140–146

Zigbee-Alliance (2006) ZigBee specification. http://www.zigbee.org/

Anis Koubâa was born in 1977 and is currently an Assistant Professor at Al-Imam Muhammad Ibn Saud University (Riyadh, Saudi Arabia) in the College of Computer Science and Information Systems and a Research Associate at the CISTER/IPP-HURRAY Research Group (Porto, Portugal). He is actively working on the design of real-time and reliable architectures for wireless sensor networks. He has driven the research efforts in the context of the ART-WiSe and open-ZB research frameworks that have contributed to the release of an open-source toolset of the IEEE 802.15.4/ZigBee protocol stack. He received his Engineering degree in Telecommunications (2000) from the Higher School of Telecommunications in Tunis (Tunisia), MSc (2001) and PhD (2004) degrees in Computer Science from the National Polytechnic Institute of Lorraine (INPL) in Nancy (France) and associated to the INRIA/LORIA research laboratory. His main research activities focus on wireless ad-hoc and sensor networks, real-time distributed systems and networks, quality of service, WPANs/WLANs integration. His is actively participating as a reviewer or a program committee member in some reputed international journals, conferences and workshops dealing with real-time networks, quality of service, wireless communications and related issues.

André Cunha was born in 1968 and has a Degree (1991), a MSc (1995) and a PhD (2003) in Electrical and Computer Engineering at the University of Porto, Portugal. He is a Professor in Electrical and Computer Engineering at the School of Engineering of the Polytechnic Institute of Porto (ISEP/IPP) and a Research Associate of the CISTER/IPP-HURRAY Research Unit. He has participated in several national and international projects (e.g. ESPRIT, IST), served in the reviewing, PC and organization of several world-reputed conferences (e.g. ECRTS) and has published tens of scientific paper at top-level conferences and journals in his expertise areas. His PhD work addressed the design of real-time hybrid (wired/wireless) factory-floor networks, in line with the RFieldbus IST Project (http://www.hurray.isep.ipp.pt/RFieldbus). Currently, his research interests are devoted to large-scale networked embedded systems, namely on supporting Quality-of-Service (QoS) in wireless sensor networks using standard protocols and Commercial-Off-The-Shelf technologies. He is leading the ART-WiSe (http/www.hurray.isep.ipp.pt/ART-WiSe) and open-ZB (http://www.open-ZB.net) research frameworks.

Mário Alves was born in 1967 and has received the Licentiate, MSc and PhD degrees in electrical and computer engineering from the University of Porto, Porto, Portugal, in 1990, 1995 and 1999, respectively. Currently he his Professor of Industrial Computer Engineering in the Computer Engineering Department at the Polytechnic Institute of Porto (ISEP-IPP), where he is also engaged in research on real-time distributed systems, wireless sensor networks, multiprocessor systems, cyber-physical systems and industrial communication systems. He heads the CISTER/IPP-HURRAY Research Unit (UI 608), a top ranked (”Excellent”) unit of the FCT Portuguese network of research units. Since 1991 he authored or co-authored more than 100 scientific and technical papers in the area of real-time computing systems, wireless sensor networks, distributed embedded systems and industrial computer engineering. Eduardo Tovar has been consistently participating in top-rated scientific events as member of the Program Committee, as Program Chair or as General Chair. Examples are: IEEE RTSS (Real Time Systems Symposium); IEEE RTAS (Real-Time and Embedded Technology and Applications Symposium); IEEE SDRS (Symposium on Distributed Reliable Systems); IEEE ICDCS (International Conference on Distributed Computing Systems); ACM EMSOFT (Annual ACM Conference on Embedded Software); Euromicro ECRTS (Euromicro Conference on Real-Time Systems); IEEE ETFA (Emerging Technologies on Factory Automation) or IEEE WFCS (Workshop on Factory Communication Systems). He has been Program Chair for ECRTS 2005 and WDES 2006 and Program Co-chair of OPODIS 2007.

Eduardo Tovar was born in 1981 in the city of Porto, Portugal. He has a degree in Computer Science from the School of Engineering of the Polytechnic Institute of Porto (2005) and an MSc degree in Computer Networks and Services from the Faculty of Engineering University of Porto (2007). He is a full time research assistant of the IPP-HURRAY!/CISTER Research Unit since February 2005. His research interests are in the thematic areas of wireless sensor network, critical systems, reliable and real-time communications, large-scale embedded distributed systems and ubiquitous computing. His MSc thesis focused on the use of IEEE 802.15.4/ZigBee as federating communication protocols for Wireless Sensor Networks, building upon an open-source implementation of the IEEE 802.15.4/ZigBee protocol stack in nesC/TinyOS for the TelosB and MICAz mote platforms (available at http://www.open-zb.net).