Distributed Scheduling of Enhanced Beacons for IEEE802.15.4-TSCH Body Area Networks

Mengchuan Zou[1], Jia-Liang Lu[1], Fan Yang[1], Mathilde Malaspina[2], Fabrice Theoleyre[3](✉), and Min-You Wu[1]

[1] Department of Computer Science and Engineering, Shanghai JiaoTong University, Shanghai, China
[2] Department Télécommunications, INSA-Lyon, Universite de Lyon, Lyon, France
[3] ICube, CNRS/University of Strasbourg, Strasbourg, France
theoleyre@unistra.fr

Abstract. Body Area Networks (BANs) expect to exploit IEEE802.15.4-2015-TSCH, proposing an efficient MAC layer for wireless industrial sensor networks. The standard relies on techniques such as channel hopping and bandwidth reservation to ensure both energy savings and reliable transmissions. With the expected growth of the BAN usage, we must now consider dense topologies, and interference. In this paper, we propose a rescheduling algorithm to avoid the collisions among the Enhanced Beacons (EB): each coordinator is able to adapt distributively its transmission to avoid interference. Indeed, EB losses impact negatively the performance of a BAN. We also optimized conjointly the neighbor discovery mechanism since a multichannel MAC would else increase too much the discovery delay. Our simulations validate the relevance of our discovery and scheduling mechanisms to cope with a very dense deployment of interfering BANs.

1 Introduction

BANs consist in small wearable wireless devices expected to fulfill the society needs in a variety of applications such as ubiquitous monitoring, health-care, entertainment and multimedia, and training. The PAN coordinator often collects measures transmitted from the wearable devices in its BAN.

The IEEE802.15 working group is currently finalizing the IEEE802.15.4-2015 standard [1]. In particular, the TSCH mode aims at improving the reliability for industrial sensor networks in noisy environments. A common schedule aims at reserving a certain amount of bandwidth for each flow, and channel hopping aims at defeating narrow band noise. This standard is particularly accurate for Body Area Networks, where devices aim at transmitting periodically their measures to the PAN coordinator (i.e. BAN gateway).

However, IEEE802.15.4-TSCH was originally designed for a large multihop wireless sensor network. In Body Area Networks, we rather face to a user centric topology: one PAN coordinator attached to a collection of devices. However, because we expect to have a large collection of co-located BANs, we will face to an explosion of collisions: each BAN computes independently its own schedule.

© Springer International Publishing Switzerland 2016
N. Mitton et al. (Eds.): ADHOC-NOW 2016, LNCS 9724, pp. 3–16, 2016.
DOI: 10.1007/978-3-319-40509-4_1

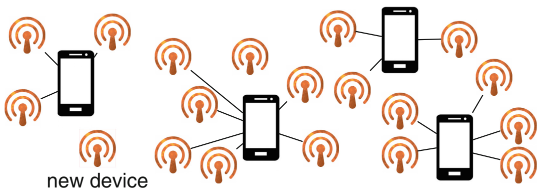

Fig. 1. Multi-BAN topology

We must propose a certain cooperation among different BANs to detect collisions and to adapt locally their schedule. Besides, BANs require a very short network attachment delay: a person may exchange data sporadically within a group of individuals (aka. opportunistic mobile social networks [2]). Thus, we must propose a mechanism to detect quickly neighboring BANs.

Due to the dynamic environment for BAN communication, one important assumption is that a newly arrived node has no initial information about the neighborhood. Therefore it needs to set up a listening schedule determining when, for how long, and on which channel it should listen to the beacons. When it receives an Enhanced Beacon (EB), it is then able to join the corresponding BAN. We chose here to optimize this discovery delay, i.e. waiting time before receiving the first EB.

In our scenario, we consider co-located BANs, possibly mutually interfering (Fig. 1). However, we consider here only single hop traffic: only neighboring PAN coordinators /devices may exchange data packets. We have consequently a collection of stars, independent concerning the traffic, interdependent concerning interference.

The contributions of this paper are fourfold:

1. we introduce a cooperation among independent BAN: while they don't exchange traffic, we propose mechanisms to locally re-schedule the transmissions of beacons to limit the number of collisions;
2. we reduce the schedule problem (and the detection of collisions) to a classical maximum-weight independent set problem in undirected graphs;
3. we also propose a mechanism so that a new node is able to discover quickly an existing neighboring BAN, even when its EBs use channel hopping;
4. we highlight the relevance of our approach with omnet++ simulations.

2 Related Work

2.1 IEEE802.15.4-2015

IEEE802.15.4-2015 [1] has proposed the TSCH mode for industrial wireless sensor networks. To improve the reliability while maximizing energy savings, a schedule is computed by the Path Computation Engine (PCE) and distributed to all the nodes. The schedule is then repeated periodically, and the ASN (Absolute Sequence Number) counts the number of slots since the beginning.

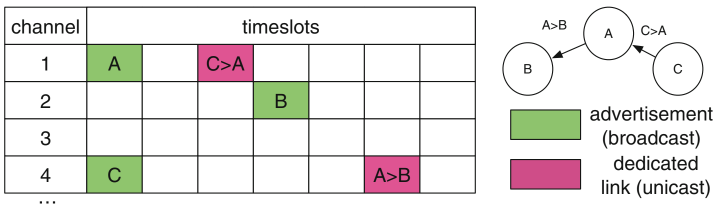

Fig. 2. Schedule in a IEEE802.15.4-TSCH network – illustration of a slotframe (Color figure online)

When a timeslot starts, a device knows if it has to wake-up to transmit or receive a packet. A slot can be either dedicated (without contention) or shared (a CSMA-CA mechanism handles contentions). Thus, a node turns off its radio for all its unused slots, where it is neither receiver nor transmitter.

To improve the reliability, TSCH proposes to implement a channel hopping scheme. To each transmission opportunity is attached a *channel offset*. Let's consider the schedule illustrated in Fig. 2. Three timeslots/channel offsets are dedicated for the advertisements of the nodes A, B and C (in green). Two other slot/channel offsets are dedicated for some radio links (in pink), for their unicast transmissions.

Palatella *et al.* proposed a centralized scheduling for a multihop IEEE802.15.4-TSCH [3] for unicast transmissions. Accetura *et al.* also proposed a decentralized version of their scheduling algorithm [4]. However, these approaches only work within a single BAN, i.e. they don't address the case of multiple interfering and concurrent BANs.

2.2 Neighbor Discovery

Two major methods exist to discover a neighboring node:

Passive discovery: a new node has to listen for the `beacon` packets from its neighbors. The period of `beacon` transmissions directly impacts the discovery delay.

Active discovery: a new node sends an `hello` packet, and neighbors acknowledge it to notify the transmitter of their presence. In multichannel, it is related to the birthday protocol [5]. This method often performs faster, but consumes more resource, and may create collisions with data packets.

A new device has to discover very fast a BAN to attach to. However, multichannel increases the convergence delay, since a receiver is deaf to the transmissions on different channels. In Bluetooth, a *transmitter* sends avertissements and a *listener* must receive this advertisement before exchanging packets. Thus, a node must change randomly its role between listener and transmitter for Peer-To-Peer topologies [6].

In multichannel, scanning all the channels takes a long time. Dasilva *et al.* proposed to use a complete permutation of all possible channels to scan, which decreases the convergence delay [7]. When a group of nodes can cooperate, they can exchange information to accelerate the discovery [8,9].

IEEE802.15.4 proposes a passive discovery: a new device sequentially scans each channel for a sufficient long time. The scanning time should be sufficient to capture any Enhanced Beacon from a neighbor. A device may require in the worst case 8.7 h to scan the 16 channels. To accelerate the discovery time, Karowski *et al.* presented a linear programming model describing the discovery process [10]. Later, they constructed also an optimal schedule, to discover in priority the nodes with a larger duty cycle ratio [11]. However, they focus on the listening device, so that their optimization could only be applied to a fixed advertising schedule.

3 Collision-Free Schedule of Multiple BANs

We consider a dense collection of Body Area Networks, with one PAN coordinator per BAN. Any pair of nodes may interfere with each other, and create collisions. We adopt in this paper the notation described in Table 1.

3.1 Problem Statement

Each BAN may select a different duty cycle ratio. In particular, they may choose a different slotframe length (the cycle duration of the schedule). Each node sends an Enhanced Beacon (EB) at a fixed interval during an advertisement slot of the schedule. More precisely, a node u sends an EB every:

$$T_{EB}(u) = T_{cst} * 2^{BO} \tag{1}$$

where T_{cst} is a duration defined by IEEE802.15.4-2015, and BO is the beacon order (constant) selected by the node u (the rest of the notation is described in Table 1).

IEEE802.15.4-TSCH adopts a FTDMA approach. We consider the PAN coordinator is co-located with the Path Computation Engine (PCE). It computes

Table 1. Notation used in the article

Notation	Meaning
$T_{EB}(u)$	the interval between two Enhanced Beacons (EB) of the node u (denoted further EB interval)
$slot_{EB}(u)$	the cumulative nb. of broadcast slots (EBs) for the node u since the beginning of the scan
nb_{ch}	nb. of channels
nb_{tslots}	nb. of timeslots for all the possible channel offsets
ASN	Absolute Sequence Number (≈time)
T_{scan}	Duration of the scan of the discoverer (nb. of slots)
$nb_{rx}(EBs)$	nb. of EB received during the scan by the discoverer
$\mathcal{S} = <(rl,t,c)>$	the IEEE802.15.4-TSCH schedule $\mathcal{S}$ is a set of triplets <radio_link, timeslot, channel_offset>

which channels and timeslots have to be used for all the pairs of active nodes, and distributes this schedule. In particular, it assigns an advertisement slot for each node to transmit periodically its EBs.

The PAN coordinator changes dynamically the schedule when it detects a collision with a neighboring network.

A new device has to receive the Enhanced Beacons to decide which network it should join. We adopt consequently a similar objective as [11]: we aim at maximizing the number of EB $nb_{rx}(EBs)$ received during a scanning duration $T_{scan}/nb_{rx}(EBs)$. All the PAN coordinators have to compute a schedule for their network, where the number of collisions between EBs is minimized.

We consider the different BANs are able to maintain a synchronization, so that all the schedules are *aligned*. It may be implemented by maintaining a synchronization with neighboring BA, such as [12] does.

3.2 Problem Formulation

We use the conflict graph model to define an accurate schedule. Let $G_c = (V_c, E_c)$ be an undirected graph, where V is the set of radio links, and E is the set of mutually interfering radio links. G_c is named the *conflict graph*. The schedule consists in the set of activated radio links $v = (rl, t, c) \in V_c$, where rl is the radio link, t a timeslot, and c a channel offset (cf. Fig. 3). For the sake of simplicity, a radio link may also be denoted by the transmitter only if it corresponds to a broadcast (i.e. EB).

An edge $(u, v) \in E_c$ indicates transmitters u and v interfere with each other:

1. either both transmitters own to the same BAN. The PAN coordinator can easily reallocate the transmitters to use different timeslots/ channels;
2. or both transmitters own to different BAN: the corresponding PAN coordinators must cooperate to avoid this kind of overlap.

Let's consider the nodes u and v with their associated beacon periods ($T_{EB}(u)$ and $T_{EB}(v)$). The pair of nodes should not share any common timeslot and channel for *any* of their Enhanced Beacons. If they have different slot frame lengths, a collision may occur if they have the same channel offset and a common active slot to advertise their EB during at least *one* slotframe:

$$\exists (i_v, i_u) \in \mathbb{N}^2 \; / \qquad slot_{EB}(v) + i_v * T_{EB}(v) = slot_{EB}(u) + i_u * T_{EB}(u) \quad (2)$$

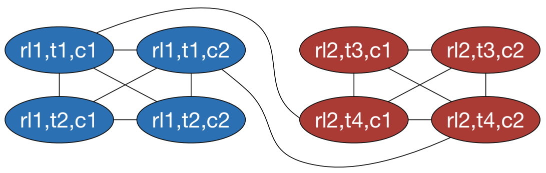

Fig. 3. Conflict graph model for ASP

This equation is an indefinite equation, with a solution (i_u, i_v). We may use the Extended Euclidean algorithm to check the existence of an integer solution. We are then able to construct the conflict graph, with all the edges which respect the Eq. 2.

3.3 Optimization Function

A node may attach to the network as soon as it discovers an accurate neighbor. Let $w(u)$ be a weight associated to a node u, denoting the amount of EBs it transmitted during the scanning time:

$$w(u) = \frac{T_{scan} - slot_{EB}(u)}{T_{EB}(u)} \tag{3}$$

Our objective consists in maximizing the total amount of EBs received by all the devices during the scanning period:

$$Objective = max\left(\sum_{u \in V} w(u)\right) \tag{4}$$

3.4 Reduction to the Maximum-Weight Independent Set Problem

When no collision occurs, the number of received EBs equals to the number of sent EBs. More precisely, no edge is present in the conflict graph between any pair of vertices: they form an independent set.

Let us consider the undirected graph $G_c = (V_c, E_c)$. An Independent Set (IS) is a set of vertices such that no edge exists between any pair of vertices. Let $\mathbb{S}$ be the set of all possible independent sets:

$$S \in \mathbb{S} \Leftrightarrow \forall (u, v) \in V_c^2 \cap S^2, \nexists (u, v) \in E_c \tag{5}$$

If the set of active vertices in G_c (i.e. radio links) forms an independent set, no collision arises: the schedule of EBs is optimal. Let $\mathbb{S}$ be the set of all possible independent sets:

$$if\ S \in \mathbb{S},\ Objective = \sum_{u \in S} w(u) \tag{6}$$

Since no conflict exists in the conflict graph, the maximization objective can be removed safely: no EB is dropped because of collisions. In conclusion, the optimization problem turns out to be a maximum-weight independent set problem (MWIS) in a undirected graph. We schedule then a MWIS in a given timeslot/channel offset.

MWIS consists in finding an independent set $S \subseteq V$, which has the maximum weight:

$$max\left(\sum_{v \in S} w(v)\right) \tag{7}$$

As we aim at maximizing the number of EBs received, our problem may be reduced to the MWIS problem, known as an NP-hard problem [13].

4 Enhanced Beacon Advertising Scheduling (EBAS)

We propose here a localized heuristic. Indeed, a centralized approach presents the following limits:

1. Overhead and Consistency: An exhaustive search requires a global knowledge. All the schedules owning to different BANs must be shared. Practically, a huge volume of information has to be exchanged, and inconsistencies may arise because of packet losses;
2. Complexity: An exhaustive exploration of the solutions is unrealistic for devices with limited capabilities. Besides, we cannot assume a server (connected to the PAN coordinators) is dedicated to this computation.

Our algorithm proceeds in the following way:

Step 1 - Division: we first create groups of devices, sharing the same EB period. We can assign orthogonal resources (i.e. channel offsets) to different groups to avoid collisions. This way, we avoid collisions among devices with different EB periods;

Step 2 - Initialization: a coordinator selects an initial schedule for all the EBs in its BAN. Each device maintains the list of occupied cells in its neighborhood, to detect collisions (i.e. at least *one* interfering device exists);

Step 3 - Collision resolution: we propose heuristics to re-schedule the colliding slots. We use hash tables to detect collisions and we re-allocate the concerned timeslots while limiting the number of changes in the schedule;

Step 4 - Backtracking: if the algorithm does not succeed to allocate a timeslot for each EB, we backtrack to the step 1, by selecting a group with a larger EB period.

4.1 Division: Dealing with Different EB Periods

We consider the same conflict graph as previously: a vertex is the triplet <*device, timeslot, channel_*offset>, and two vertices are linked together if the associated devices interfere.

We can remark the following properties:

1. All vertices representing the same device (for different EBs) form a clique in the conflict graph;
2. when considering a set of vertices with the same EB interval and using the same channel offset, the collision is present for *all* the timeslots, and can be easily detected.

So, we propose a *partition and re-grouping* method:

1. we divide the graph into cliques. We insert an edge between two cliques if they are neighbors in the conflict-graph;
2. we merge the cliques with the same EB interval into a group. We will assign later a given channel offset to a group;

3. then, we assign a given timeslot for each device within a group: they should not interfere since they use the same channel offset. Assigning distinct timeslots is sufficient since they have the same T_{EB}.

By adopting a three-steps approach, we are able to deal with devices with different EB intervals. Indeed, devices with different intervals are partitioned in distinct groups (i.e. different channel offsets). Then, we solve the conflicts among the devices with the same EB interval, using a TDMA repartition. Since times slots are distinct, they cannot collide in *any* slot frame: the period of repetition is the same for all of them – same T_{EB}.

Group Merging. If too many EB intervals exist, partitioning the graph would create too many groups. However, as Eq. (1) states, the EB intervals are multiples of 2. Let's denote by $group_i$ all the devices with an EB period equals to BO^i.

Several devices of the group i may actually be scheduled with the devices in the group $i - 1$. The devices of the group i have either to be scheduled in different timeslots or in different slotframes. Intuitively, a slotframe of the group i contains *two* slotframes of the group $i - 1$.

Obviously, we can merge any group i and j $(i < j)$ of devices. Finally, at most $2^{BO^j - BO^i}$ nodes of the group j may be schedule in the final same timeslot. Inversely, a device of the group i uses a given timeslot in *all* the slotframes.

4.2 Initialization: Assigning Timeslots to Each Device

All the devices which share the same EB interval are scheduled in the same channel offset, and the PAN coordinator assigns randomly one free timeslot for each of them. The PAN coordinator has to compute its schedule, and to push it to all the devices. The schedule table is actually the set of timeslots/channel offsets (row), and their corresponding occupation (0 if this timeslot is unoccupied by any transmitter).

Each PAN coordinator broadcasts its schedule periodically in dedicated packets, with an increasing sequence number. A neighbor is thus able to detect a timeslot collision, equivalent to a collision in both hash tables.

A node is able to maintain an up-to-date schedule table of its neighboring PAN coordinators:

- any neighbor for which it stops receiving packets is removed from the neighborhood table after a certain timeout;
- only the last schedule is memorized for each neighbor (i.e. largest sequence number).

Thus, we guarantee a certain consistency in the decisions: each device is able to replace obsolete information from its neighborhood schedule table.

4.3 Collision Resolution in the Same Group - Linear Time Scanning

A PAN coordinator may detect a collision after the reception of the schedule from one neighbor. To solve the collisions, we use hash tables.

More precisely, an hash table makes a correspondence between a key (here, a timeslot) and the value (1 if the timeslot is occupied by at least one node, else 0). A collision of timeslots means also a collision in the hash table. Thus, finding compliant schedules (i.e. interference free) is equivalent to resolving the collisions in the hash table.

Two methods exist to resolve collisions in hash tables:

1. separate chaining: each entry of the has table is a chained list of values;
2. open addressing: the values are stored in contiguous entries in the hash table.

Since the former approach does not maintain the original order in the hash tables, we adopt the open addressing method.

When the network is very dense, many timeslots are occupied. To accelerate the convergence, we propose here to solve several conflicts simultaneously. The following two steps are required:

1. Preprocessing: for any channel, the node combines all the schedules present in the neighborhood table with the OR operator. Then, the node is able to compute the list of slots used by at least one of the BANs, and the list of the idle slots.
2. Linear time scan: we create two pointers (cf. Fig. 4). The first one (in red) points to the first colliding slot in our schedule, and the second one (in green) to the first idle slot in the merged schedules. Then, we modified our schedule: the colliding cell is moved to be placed in the next free cell (in the merged schedule). Then, the pointers keep on iterating, to solve the next conflict.

When resolving the conflicts, both pointers visit each slot exactly once. Thus, our algorithm runs in $O(n)$ time. We have consequently a linear time rescheduling algorithm.

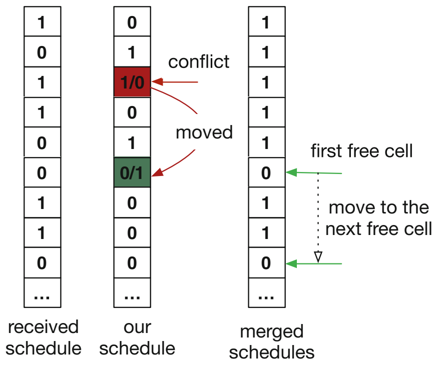

Fig. 4. Linear time scanning (Color figure online)

4.4 Backtracking: Group Change

If the previous search has failed, this means no timeslot is available.

We backtrack the algorithm to the step 3. The device joins a group with a lower EB interval, and re-executes the assignment to find an idle slot in the new group.

This backtracking method is particularly relevant when all the groups do not have the same number of devices. A BAN will be moved to another group to balance the load.

4.5 Analysis

Let's now consider the initialization step. We have nb_{tslots} slots (all the timeslots in a slotframe, across the nb_{ch} possible channel offsets). The number of EBs to be sent is n, and timeslots are assumed to be assigned randomly for initialization.

So, the expected number of collision-free EBs (n_{nocoll}) is the complementary of colliding EBs:

$$n_{coll} = n * \left(1 - \prod_{k=2}^{n} \frac{nb_{tslots} - (k-1)}{nb_{tslots}}\right) \tag{8}$$

We denote by ρ the ratio of the number of devices and the number of timeslots ($\rho = n/nb_{tslots}$). Thus ρ denotes the *pressure* for the scheduling process. The ratio σ of conflicting EBs is consequently:

$$\sigma = \frac{n_{coll}}{n} = 1 - \prod_{k=2}^{n} \frac{n - \rho * (k-1)}{n} \tag{9}$$

Because of the well-known birthday problem [14], the ratio of collisions will be quite large when ρ grows.

Our algorithm solves the conflicts created in the initialization phase. We group the devices by their EB interval, and thus their channel offset. We denote by x_i the number of devices using the channel offset i. If the following condition holds, our scheduling algorithm is able to assign non colliding slots:

$$\forall i \in [0..nb_{ch}],\ x_i \leq \frac{nb_{tslots}}{nb_{ch}}, \tag{10}$$

In other words, there exist enough timeslots to schedule all the devices of a group.

According to [15], our algorithm converges with the probability:

$$P\left[\forall i, x_i \leq \frac{nb_{tslots}}{nb_{ch}}\right] = 1 - nb_{ch} * \binom{n}{\frac{nb_{tslots}}{nb_{ch}}} * (\frac{1}{nb_{ch}})^{\frac{nb_{tslots}}{nb_{ch}}} \tag{11}$$

This represents a lower bound for the probability of convergence. For each iteration, we have:

1. After probing and changing a schedule on one channel i, we have $x_i^{t+1} \leq x_i^t$ which means x_i doesn't increase.
2. If the device switches to another channel j, we have

$$x_i^{t+1} \leq x_i^t - 1, x_j^{t+1} \leq \frac{nb_{tslots}}{nb_{ch}} \tag{12}$$

so that there is also $\forall i, x_i \leq \frac{nb_{tslots}}{nb_{ch}}$.

Moreover, if $\exists i, x_i > \frac{nb_{tslots}}{nb_{ch}}$, at least one device is able to switch to another channel offset to find a free timeslot. So Eq. 11 is a lower bound of convergence probability.

5 Performance Evaluation

We used Castalia 3.0 (http://castalia.npc.nicta.com.au), a simulator based on the OMNeT++ framework version 4.1 (http://www.omnetpp.org). Castalia is widely used to simulate a Body Area Network. For the initialization, each coordinator chooses independently and randomly a timeslot to send EBs.

We simulated a multi-BANs scenario where a set of PAN coordinators are directly connected to a random number of devices. The number of coordinators varies between 5 and 20 (with increment of 5). Besides, the ratio of slot occupation varies from 10 % to 90 %, with increment of 20 %. We run 500 simulations for each set of parameters.

We compared the following algorithms:

- `PSV`: the passive scan mode in IEEE802.15.4-2015 [1]. An arriving device just listens for the maximal possible length of time on each channel;
- `SUBOPT`: this low-complexity algorithm computes a listening schedule [10]. Intuitively, the device has not to scan all the timeslots when a neighbor sends very fast its EBs. It avoids these *redundant* timeslots;
- `EBAS`: our rescheduling algorithm detecting collisions and reallocating the colliding devices to free timeslots.

We have the following main parameters/performance criteria:

- ratio of occupation: the ratio of the number of devices (EB to schedule) and the number of timeslots;
- Percentage of EBs received: number of EB received by a discovering device. The higher this value is, the faster a device may discover a network to join;
- Percentage of EB conflicts: ratio of EB which use a timeslot which collides with another EB (i.e. for at least one timeslot when both transmitters have not the same EB interval).

We first measured the number of EBs correctly received (Fig. 5). The efficiency of SUBOPT and PSV decreases when the network comprises more coordinators: SUBOPT has been designed to reduce the discovery delay, but doesn't

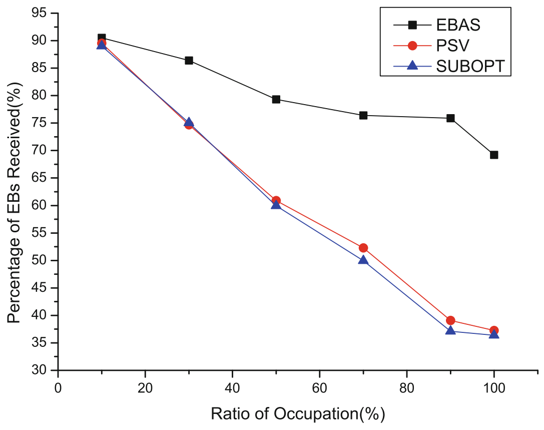

Fig. 5. Ratio of EBs received correctly (i.e. without collision)

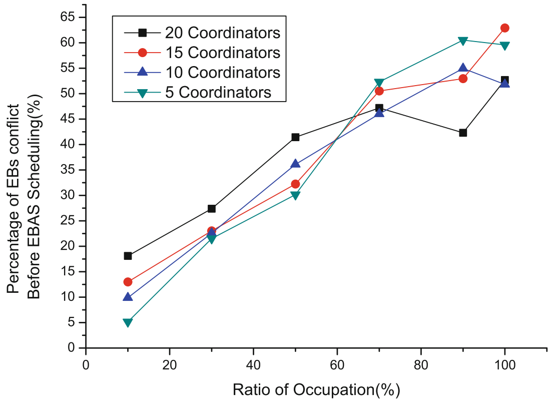

Fig. 6. Ratio of colliding EBs before EBAS algorithm

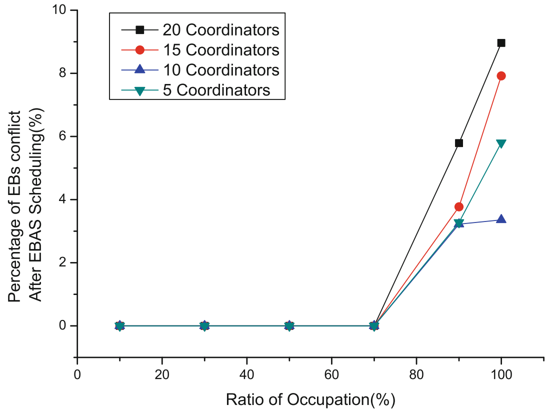

Fig. 7. Ratio of colliding EBs after EBAS algorithm

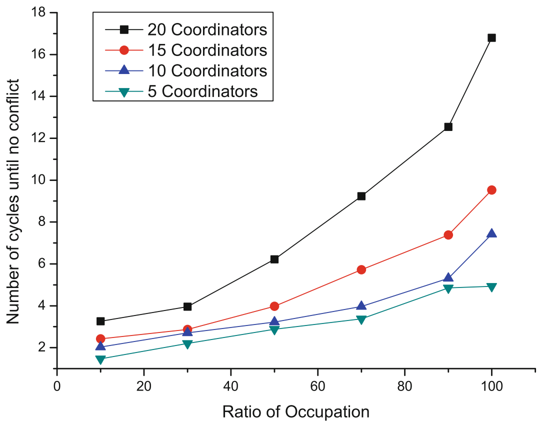

Fig. 8. Number of iterations needed until a conflict-free schedule is found

try to re-schedule the colliding beacons. On the contrary, EBAS is much more scalable and successfully solves most of the conflicts even for a very large occupation ratio. Such feature is vital for very dense BAN deployments.

We also measured the percentage of colliding EBs without (respectively with) EBAS in Fig. 6 (resp. Fig. 7). Comparing both figures, we clearly notice EBAS is very efficient when less than 70 % of the slots are occupied: it solves 100 % of the collisions, re-allocating the incriminated devices. Even for a very large occupation ratio (e.g. 90 %), only 10 % of the EBs collide. On the contrary, a distributed random assignment rather leads to a ratio of 60 % of collisions.

Finally, we measured the number of iterations with EBAS before obtaining a conflict-free schedule (Fig. 8). For small BANs, only a few iterations are required (at most 4). Without surprise, the number of iterations grows with the occupation ratio. However, EBAS converges in less than 20 iterations, even in very extreme conditions.

6 Conclusion

We proposed here an algorithm to schedule the EBs transmissions when several Body Area Networks co-exist and mutually interfere. We proposed first a dividing strategy, grouping together the devices with the same EB. Then, we propose to detect and solve collisions after a random assignment. Neighboring BANs exchange their schedule to detect conflicts, and reallocate the timeslots while preserving the other collision-free schedules. Simulations prove the relevance of our approach: we reduce the number of collisions even in dense deployments.

In the future, we plan to combine a fast collision detection mechanism with our re-scheduling algorithm. We also aim at validating experimentally our solutions with complex radio propagation (and interference). Besides, we also aim at quantifying the impact of clock drifts on the accuracy of our re-scheduling process. Finally, we expect to investigate the performance of the proposed solution under different conditions (e.g. multi-hop traffic, QoS requirements).

Acknowledgment. This research was supported by NSF of China under grant No. 61100210 and China 863 No. 2015AA015802 and by the French National Research Agency (ANR) project IRIS under contract ANR-11-INFR-016.

References

1. IEEE Standard for Local, metropolitan area networks-Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer. IEEE Std 802.15.4e-2015 (2015)
2. Pietiläinen, A.-K., et al.: Mobiclique: middleware for mobile social networking. In: ACM Workshop on Online Social Networks, pp. 49–54 (2009)
3. Palattella, M.R., et al.: On optimal scheduling in duty-cycled industrial IoT applications using IEEE802.15.4e TSCH. Sens. J. IEEE **13**(10), 3655–3666 (2013)
4. Accettura, N., et al.: Decentralized traffic aware scheduling for multi-hop low power lossy networks in the internet of things. In: IEEE WoWMoM (2013)
5. McGlynn, M.J., Borbash, S.A.: Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks. In: Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing, pp. 137–145. ACM (2001)
6. Drula, C., Amza, C., Rousseau, F., Duda, A.: Adaptive energy conserving algorithms for neighbor discovery in opportunistic bluetooth networks. IEEE J. Sel. Areas Commun. **25**(1), 96–107 (2007)
7. DaSilva, L.A., et al.: Sequence-based rendezvous for dynamic spectrum access. In: New Frontiers in Dynamic Spectrum Access Networks (DySPAN), pp. 1–7. IEEE (2008)
8. Chen, L., et al.: Group-based discovery in low-duty-cycle mobile sensor networks. In: IEEE SECON, pp. 542–550. IEEE (2012)
9. De Nardis, L., et al.: Role of neighbour discovery in distributed learning and knowledge sharing algorithms for cognitive wireless networks. In: ISWCS, pp. 421–425. IEEE (2012)
10. Karowski, N., et al.: Optimized asynchronous multi-channel neighbor discovery. In: IEEE INFOCOM, April 2011
11. Karowski, N., et al.: Optimized asynchronous multichannel discovery of IEEE 802.15. 4-based wireless personal area networks. IEEE Trans. Mob. Comput. **12**(10), 1972–1985 (2013)
12. Lipa, N., Mannes, E., Santos, A., Nogueira, M.: Firefly-inspired and robust time synchronization for cognitive radio ad hoc networks. Comput. Commun. **66**, 36–44 (2015)
13. Trevisan, L.: Inapproximability of combinatorial optimization problems. The Computing Research Repository (2004)
14. Mathis, F.H.: A generalized birthday problem. SIAM Rev. **33**(2), 265–270 (1991)
15. Borodin, A., et al.: A time-space tradeoff for sorting on a general sequential model of computation. In: ACM Symposium on Theory of Computing. ACM (1980)