Ad Hoc Networks 49 (2016) 58–69

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

A learning approach to decentralised beacon scheduling

C. Cano[a,*], D. Malone[b]

[a] Inria Lille-Nord Europe, 40 Avenue Halley, 59650, Villeneuve d'Ascq, France
[b] Hamilton Institute, Maynooth University, Co. Kildare, Ireland

ARTICLE INFO

Article history:
Received 21 January 2016
Revised 14 June 2016
Accepted 15 June 2016
Available online 16 June 2016

Keywords:
Beacon scheduling
Decentralised constraint satisfaction
Collision-free operation

ABSTRACT

Beaconing is usually employed to allow network discovery and to maintain synchronisation in mesh networking protocols, such as those defined in the IEEE 802.15.4e and IEEE 802.11s standards. Thus, avoiding persistent or consecutive collisions of beacons is crucial in order to ensure correct network operation. Beacons are also used in receiver-initiated medium access protocols to advertise that nodes are awake. Consequently, effective beacon scheduling can enable duty-cycle operation and reduce energy consumption. In this work, we propose a completely decentralised and low-complexity solution based on learning techniques to schedule beacon transmissions in mesh networks. We show the algorithm converges to beacon collision-free operation almost surely in finite time and evaluate converge times in different mesh network scenarios.

1. Introduction

Beacon transmissions are a fundamental mechanism used to maintain synchronisation and to enable network discovery in several wireless mesh networks, such as those based on the standards IEEE 802.15.4e [1] and IEEE 802.11s [2]. These control messages are normally sent at fixed time intervals. Hence, given the random access nature of these protocols, collisions of beacons can persistently occur when several devices share the medium, especially in presence of hidden terminals. That is, in the case more than one node select the same time to send the beacon, they will collide indefinitely if no mechanism to recover from these collisions is implemented. Without successfully receiving these beacons, a node is neither able to remain synchronised, nor to discover neighbouring nodes. Thus, coordinating beacon transmissions to avoid collisions is a primary concern in mesh random-access networks so as to enable effective communication among neighbouring devices [3]. Additionally, receiver-initiated protocols for Wireless Sensor Networks (WSNs), like RI-MAC [4], can benefit from beacon coordination to provide efficient broadcast support. Coordinating sensor nodes to wake up at approximately the same time removes the need to send repetitions of broadcast messages to every receiver. Therefore, this coordination will allow for a reduction in the energy consumption [5].

The standards IEEE 802.15.4e [1] and IEEE 802.11s [2] define mechanisms to alleviate collisions of beacons among neighbouring nodes. However, these approaches do not completely prevent collisions. This is especially relevant in IEEE 802.15.4e [1], where successive collisions can take place if, for instance, any of the control messages used to reserve a slot for beacon transmission are lost, as studied in [6]. In IEEE 802.11s [2], this problem is not as severe since a mechanism to periodically and randomly delay beacon transmissions is proposed to resolve collisions. However, it is still possible to experience beacon collisions for periods of time. Moreover, randomly delaying beacon transmissions is detrimental for stations working in power saving mode as they only wake up at the expected time to receive a beacon.

In this work, we propose a fully decentralised, generic and low-complexity solution to completely avoid beacon collisions based on learning. By taking a learning approach, we are able to provide scheduling without using extra control messages to coordinate neighbouring nodes. Similar techniques have been studied for scheduling unicast data packets in Wireless Local Area Networks (WLANs), known as decentralised learning for collision-free operation [7–9]. We extend these approaches to beacon coordination as follows:

- We take advantage of the fact that beacons are control messages in order to include information useful for coordinating neighbouring nodes.
- Given that beacons are sent as broadcast and thus, there is normally not feedback of the correct reception of these messages by neighbouring nodes, we propose a mechanism to provide a

* Corresponding author.
E-mail addresses: cristina.cano@inria.fr (C. Cano), david.malone@nuim.ie (D. Malone).

http://dx.doi.org/10.1016/j.adhoc.2016.06.007
1570-8705/© 2016 Elsevier B.V. All rights reserved.

node with feedback about whether its previous beacons were successfully received.

The proposed approach aims to provide fast convergence to *beacon collision-free operation*[1] while relying on local information only. Quick convergence and decentralisation is important in networks where devices may be mobile to allow practical reconfiguration upon network changes. To achieve this aim, nodes probabilistically decide at each schedule (or round) which slot to use to transmit a beacon based on information transmitted in neighbouring beacons. When all neighbouring nodes provide positive feedback about a node's beacon transmission, it keeps using the same slot in the next schedules. Thus, when all nodes have found the slot in which to transmit without collision the network operates in a beacon collision-free manner. Given that beacon collisions can still happen before the network reaches the collision-free operation phase, nodes must take decisions without a complete view of the neighbourhood. Despite this constraint, we will show that the algorithm converges almost surely in finite time. Our evaluation of the convergence rate will also show that the number of schedules required to converge to beacon collision-free operation is in the order of ten schedules in different scenarios and configurations.

This article is organised as follows. In Section 2, we motivate this work by describing open issues related to beacon scheduling. Then, in Section 3, we discuss the relevant related work on decentralised scheduling. After that, in Section 4, we describe the proposed decentralised mechanism to schedule beacon transmissions. We formulate the problem as a constraint satisfaction problem in Section 5. The analysis of convergence is provided in Section 6 while the algorithm benchmarking is presented in Section 7. We discuss practical implications in Section 8 and then, we provide some final remarks.

2. Motivation

Our approach aims to provide coordination of periodic broadcast transmissions in mesh networks. However, it is motivated by the identification of the following issues:

- The possibility of facing persistent collisions of beacons in IEEE 802.15.4e making it impossible to identify neighbouring nodes and to maintain synchronisation.
- The possibility of facing consecutive collisions of beacons in IEEE 802.11s and a poor support of stations working in power saving mode.
- The inefficient transmission of broadcast messages in receiver-initiated approaches for WSNs.

These problems are explained in detail in the following subsections. We will see that the probability of beacon collision is often relatively low, so the crux of our scheme is to identify these collisions and then make a local reassignment of slots when this is detected.

It is important to emphasise that our approach aims to provide *generic* scheduling of beacon transmissions and can be adapted to other beaconing-based networks such as vehicular or power line communication networks.

2.1. Mesh WSNs based on IEEE 802.15.4e

IEEE 802.15.4 relies on beacon transmissions for network formation and synchronisation in WSNs. Previous work has addressed

[1] We denote with *beacon collision-free operation* the case in which there is no collisions of beacons. We distinguish this situation to the case where there are no collisions of data packets, which is the operation the works in [7–9] aim for.

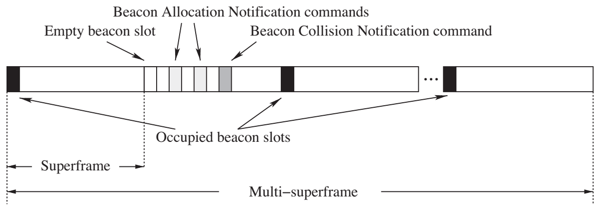

Beacon Allocation Notification commands
Empty beacon slot
Beacon Collision Notification command
...
Occupied beacon slots
Superframe
Multi−superframe

Fig. 1. Example of two nodes requesting the same slot in the multi-superframe [1].

beacon scheduling in these networks. However, many existing approaches consider a tree topology or are based on centralised solutions such as the works in [10] and [11]. Distributed solutions, like [12,13] and [3], have also been proposed for mesh WSNs and have inspired the beacon collision avoidance of the new IEEE 802.15.4e standard [1].

The Deterministic and Synchronous Multi-channel Extension (DSME) of the IEEE 802.15.4e [1] aims to support mesh networks by defining a *multi-superframe* formed by a number of superframes, each containing a beacon transmission, a contention-free and a contention access period. Each node selects a superframe in which to transmit a beacon (Fig. 1). Thus, to avoid collisions of beacons, beacon scheduling among 2-hop neighbouring nodes is needed. The standard addresses beacon coordination by including in the beacon a bitmap of the beacon allocation schedule for the neighbourhood. Therefore, a node, by receiving all beacons from its neighbourhood, is aware of the currently allocated beacons in the multi-superframe in its 2-hop neighbourhood. After that, the node selects a free slot and transmits a *beacon allocation notification command* in the contention access period of the superframe corresponding to the selected slot. If multiple nodes select the same slot, the standard relies on a *beacon collision notification command* sent from a common neighbouring node to resolve contention. This allows the last station requesting that slot to realise that a conflict has occurred (example depicted in Fig. 1).

However, DSME does not define any mechanism to recover from collisions of *beacon allocation notification commands* or *beacon collision notification commands*, as studied in [6]. The case of nodes having no neighbours in common is also not considered. In both cases, nodes assume they have gained the slot, as no negative feedback is received, and beacons will persistently collide. The consequences of this are the inability to detect neighbouring nodes and to maintain synchronisation. Thus, further design considerations are needed to schedule beacon transmissions in IEEE 802.15.4e.

In order to illustrate the magnitude of this problem, we analyse the conditional collision probability, that is, the probability that a *beacon allocation notification command* collides with another *beacon allocation notification command*. We consider a lower bound on this metric by analysing a scenario in which all nodes are in mutual coverage range (i.e., a fully connected graph) in the absence of any data transmission, thus, the only messages that are transmitted are those related to the beacon allocation process. Note that we also neglect the potential collisions among *beacon collision notification commands* when more than one *beacon allocation notification commands* are successfully received for a given subframe. Considering that a superframe is formed by 16 slots (one of them used for the beacon transmission) [14], that c slots for beacons are available (c superframes still do not have an allocated beacon) and that n nodes are attempting to allocate their beacon, the conditional collision probability is simply: $p_c = 1 - (1 - \tau)^{n-1}$, where $\tau = 1/(15c)$. Results varying n and c are shown in Fig. 2. Note that although the conditional collision probability is smaller than 2% for $n < 6$, it is still considerable given the importance of persistent collisions of beacons as outlined before. On the other hand, in all situations shown, the collision probability is $>$ 1% for $n \geq 4$ nodes.

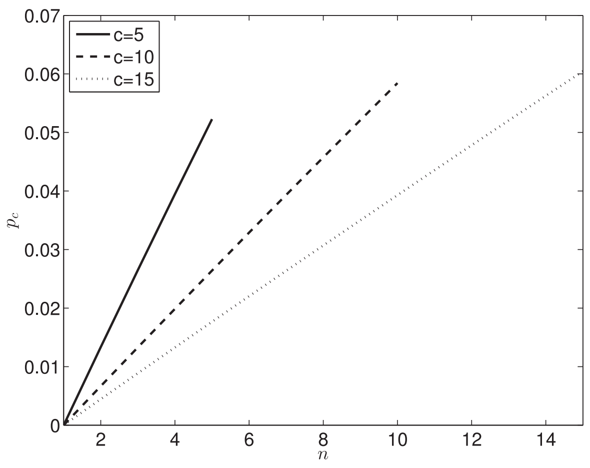


Fig. 2. Conditional collision probability of *beacon allocation notification commands*.

This means that in the case that 4 nodes or more in the neighbour try to join the schedule at approximately the same time, more than one in a hundred cases would result in persistent collisions.

Attempts to improve the standard have been presented in [6,15–17]. The approaches proposed in [6,15] are based on active associations instead of passive scans of neighbouring beacons. Thus, requiring a complete change of the initial standard design, including explicit messages for association and feedback. In contrast, in [16], the authors propose to use the active period of the superframe to transmit beacons in order to reduce the beacon collision probability, motivated by the fact that the active period is longer than the specified fraction of time devoted to transmit beacons in IEEE 802.15.4e. However, even though the collision probability of beacons is reduced in this way, beacon collision-free operation is not ensured and as in the previous approaches, a substantial change of the standard is needed, i.e., the fraction of time intended for beacon transmissions is no longer needed. The closest work to our proposal that we are aware of is [17], where the authors also propose to use the learning mechanism defined in [7] to schedule beacon transmissions. However, how a collision of beacons is detected and how the lack of information due to neighbouring beacons colliding is handled, which are central aspects in the applicability of the protocol defined in [7], are not considered. The work in [17] also does not study convergence to beacon collision-free operation in mesh networks.

2.2. Mesh WLANs based on IEEE 802.11s

The WiFi mesh standard IEEE 802.11s [2] defines the Mesh Beacon Collision Avoidance (MBCA) mechanism to alleviate beacon collisions among mesh stations. Stations use their beacons to advertise the expected time of the next beacon transmission and beacon interval of their neighbours. A node receiving this information, adjusts its own beacon transmission time so as to not overlap with other beacons in the 2-hop neighbourhood. To alleviate beacon collisions among neighbouring nodes, beacon transmissions are randomly delayed at periodic time intervals. Note that beacon collisions are not completely avoided and that a node overhearing a beacon collision is not able to get the required 2-hop neighbourhood information to adjust its own beacon transmission time. Moreover, delaying beacon transmissions is detrimental for stations working in power saving mode, which only wake up at the expected beacon reception time.

A proposal to detect collisions of beacons in IEEE 802.11s is presented in [18]. The authors propose the use of *probe* beacons sent intermittently between beacon intervals. These *probe* beacons include the time at which the beacons of the source are scheduled and so enable recipients with equal beaconing times to change their scheduled time to transmit beacons. Note that although no explicit messages for scheduling are used, extra control packets are sent in order to infer a collision of beacons taking place and that *probe* beacons are also prone to collisions.

2.3. Efficient broadcast support in receiver-initiated WSNs

In WSNs, nodes work in a duty-cycle operation in which they go to sleep and periodically wake up. Beacons are used in receiver-initiated approaches for WSNs, as the RICER [19], the RI-MAC [4] and the IRDT [20] protocols, to notify neighbouring nodes that a receiver is awake. Transmitters with data to send keep listening to the channel for the beacon from the receiver. After the beacon is received, they can start sending the data. These protocols support unicast transmission but broadcast messages are handled by transmitting a copy of the message to every receiver. Using coordination to schedule wake-up times and making just one node in the 2-hop neighbourhood send a beacon at a time improves broadcast transmission as multiple copies of the same message are no longer needed [5]. This operation can, therefore, provide further benefits for energy conservation when broadcast traffic is considered. Coordinating beacon transmissions also reduces the number of beacons sent, thus the channel is expected to saturate more gradually. How to coordinate beacon transmissions in RI-MAC in a fully-connected network was studied in [5]. In that work, it was shown that the time to convergence is low, less than 11 schedules, even for scenarios in which 60 nodes compete for 60 time slots.

3. Related work on decentralised scheduling

As far we know, this is the first work to propose and analyse a fully decentralised solution to achieve beacon collision-free operation that does not require the use of extra control messages to coordinate neighbouring nodes. However, the problem addressed here belongs to the well-known problem of decentralised resource allocation. In this section, we give an overview of the most closely related approaches: *(i)* decentralised learning for collision-free operation of unicast packets in WLANs and *(ii)* distributed solutions to schedule broadcast transmissions. Note that this work differs from scheduling of beacons for data collection in the sense that our main purpose is on reliability instead of on minimising latency in data collection, such as in [21,22]. The extension of this work to data collection applications is an interesting future research line. However minimising data collection latency is a complex problem, and would distract from our aim of presenting a general protocol for beacon collision free operation.

Applying decentralised learning for collision-free operation of unicast data packets has been studied for Wireless Local Area Networks (WLANs) in [7,8] and [9]. A schedule of transmissions that is repeated in cycles is defined, where the schedule length is the number of slots per cycle. The general idea of these mechanisms relies on the nodes randomly picking a slot in the schedule and then remain transmitting in the same slot in the subsequent cycles if their transmissions are successful (i.e., an acknowledgement is received back). In case of a collision, nodes randomly change the slot selection. The work in [7] only selects among the empty slots in the schedule. In contrast, the proposal in [8] allows nodes to pick any slot in the schedule so as to not require them to monitor the status of each slot in the previous cycle. In [9], both approaches are extended to improve the convergence rate (the time to reach collision-free operation). To achieve this goal, a learning parameter (γ) is defined. After an unsuccessful transmission, nodes change the slot selection with $(1-\gamma)$ probability and remain transmitting in the same slot with probability γ. In this work, we aim to extend these approaches to beacon transmissions. As beacons are broadcast, no acknowledgements are transmitted, so one of the main challenges is how to provide a node with feedback about successful beacon receptions at its neighbouring nodes.

Distributed algorithms for broadcast scheduling for nodes without previous knowledge of their 2-hop neighbourhood depend on joining procedures where explicit communication allows to create a collision-free schedule. Some examples are the mechanisms in [23] and [24]. In our work, no request to use a particular slot is needed as nodes probabilistically decide which slot to select and rely on indirect information from the neighbourhood as feedback to determine if a slot selection causes a conflict. Therefore, compared to previous approaches, the mechanism presented in this work reduces the overhead in dynamic scenarios. Moreover, given that the probability of beacon collision is expected to be low, as showed in Fig. 2, we could define a centralised algorithm that reschedules beacon transmissions only when a conflict is detected, thus, without incurring excessive overheard. However, such an approach would still require a mechanism to detect that a collision is taking place. What we will show in this work is that we are able to re-schedule slots in a decentralised manner using only the information necessary to detect the conflict.

4. Decentralised beacon scheduling algorithm

In this section, we provide an overview of the algorithm to schedule beacon transmissions in a decentralised manner.

4.1. Assumptions

Here we detail the assumptions used to design the algorithm. Later, in Section 8, we will discuss how they relate to practical deployments and existing standardisation efforts.

Local synchronisation: We first assume that, although no global synchronisation is required, nodes have the capability of local synchronisation. This requirement can be achieved by including synchronisation information in beacons. Using this information, nodes in the network adapt their clocks based on the synchronisation information received from neighbours.

Predefined time intervals for beacon transmissions: We assume beacons are sent at specific time intervals. In this way, a node overhearing a collision during these intervals can infer the collision corresponds to overlapping beacon transmissions instead of data packets. We consider that the predefined intervals for beacon transmissions are slotted. The slot duration must be fixed so as to avoid misalignment caused by different observed channel status at different locations.

Sufficient number of slots: In the following we assume that the number of slots in a schedule is sufficient to allow for collision-free operation of beacons in a two-hop neighbourhood.

4.2. Overview

The protocol operation is divided in two parts: *(i)* obtaining feedback from beacons sent by neighbours and *(ii)* deciding the slot to use in the next cycle (schedule) based on the feedback obtained. Both are explained in detail next.

4.2.1. Obtaining feedback from neighbouring beacons

The beacon allocation schedule is repeated in cycles, each cycle being of length $C \in \mathbb{N}$. Nodes include in beacons the value of C as well as the position that the current node occupies in the schedule (its currently selected slot for beacon transmission). Following this approach, nodes receiving this information have knowledge about the current length of the schedule and when it starts/ends.

To schedule beacon transmissions, we take advantage of the fact that these are control messages. We consider that nodes include in their own beacons feedback of previously overheard beacon transmissions that serve as implicit acknowledgements for neighbouring nodes. For this purpose, a bitmap of previously overheard beacons at each slot in the schedule is included in every beacon. Note that this is a similar approach to the one defined in IEEE 802.15.4e [1], where a bitmap of the slot occupancy is included in beacons. However, in the case addressed here, this bitmap includes information on *(i)* slots where beacons were correctly received, *(ii)* slots in which the node inferred that a collision of beacons occurred and *(iii)* slots where no beacon was transmitted. Including this extra information results in an additional bit per slot in the schedule, however, it is useful for a node in order to:

Infer a satisfactory slot selection: A node assumes its beacon was correctly received by all its neighbouring nodes if all of them signal a correct reception in its previously selected slot.

Have a complete view of the 2-hop neighbourhood: Beacons from neighbouring nodes include information of their own neighbourhood. Thus, a node is able to infer the slots used in the 1-hop (overheard beacons) and 2-hop (slots advertised as occupied by overheard beacons but seen as free by the given node) neighbourhood. This information is used to select the next slot to use in case no positive feedback is received from all neighbouring nodes in the current schedule.

Although it is possible to schedule beacons using this information, there are challenges that must be addressed in order to overcome the following limitations:

Feedback is prone to loss: The first challenge is based on the fact that some information may not be accessible. Before convergence, it is possible that beacons from neighbouring nodes collide, making impossible for a node to obtain the information included in them. In this case, a node is not certain whether its beacon was correctly received by all its neighbouring nodes. To solve this issue, after observing a collision in a slot where a beacon should be transmitted, a node in our scheme acts as if negative feedback was received.

No node to provide feedback: Another challenge is the case where there is no node to flag the occurrence of a collision. A given beacon transmission can collide with other beacon transmissions from neighbouring nodes and if transmitters do not share any neighbour to notify them of the collision, then, they are not able, in principle, to realise they are colliding. It is also possible that all nodes in a neighbourhood are colliding in the same slot. To solve this problem, we assume nodes are able to infer a full list of their neighbours, say by overhearing regular transmissions. Since nodes can still transmit data messages even if their beacon collide, this assumption is reasonable. Therefore, a node assumes it is colliding with a neighbour by noting the lack of its beacon transmission. Then, it also acts as if negative feedback on the selected slot was received.

4.2.2. Decision making based on received/lack of feedback

When a node joins the network, it keeps listening to the channel for beacon receptions from its neighbouring nodes. After listening for CT_b, with C being the initial length of the schedule and T_b the period between beacon transmissions, two situations can arise: *(i)* no beacon is received, then the node assumes it is the first node in the network and starts sending its beacon in a random (uniformly selected) slot in the schedule, or *(ii)* it receives at least one beacon from a neighbour. In the latter case, the node randomly (following a uniform distribution) selects an empty slot (a slot advertised as free by all its neighbouring nodes).

As explained in the last subsection, after sending a beacon, a node obtains feedback of this transmission from beacons sent in its neighbourhood and also from the lack of them. A node con-

Algorithm 1: Pseudocode of the beacon scheduling algorithm.

```
Data: C, $T_b$, $\gamma$
Listen for $CT_b$;
if Beacon(s) received then
    Update unused slots;
    Select slot randomly in unused slots;
else
    Select slot randomly in C;
end
for every cycle do
    Listen for Beacons;
    Update unused slots;
    if All neighbours flag correct reception then
        Continue using the same slot;
    else
        if $rand(1) < \gamma$ then
            Continue using the same slot;
        else
            Select slot randomly in unused slots;
        end
    end
end
```

siders that it has gained the slot if all the following occur: *(i)* all beacons received advertise the previously selected slot as a successful reception or as empty,[2] *(ii)* it does not overhear a collision of beacons among neighbouring nodes and *(iii)* it receives all beacons sent by the nodes in its neighbourhood. If any of those fails, the node considers it has not gained the slot and therefore, a decision about which slot to use in the next schedule must be taken.

In order to reduce the convergence time, we use the mechanism proposed in the Learning Zero Collision (L-ZC) protocol [9]. L-ZC relies on previous information of the occupied, collision and empty slots. Since we have chosen to include this information in the beacons, L-ZC is a good option for the problem addressed here. Then, after realising it has not gained the slot, the node changes to one of the slots seen as free in the last schedule with $(1-\gamma)$ probability and remains in the same slot with probability γ, where γ is a design parameter. The value of this parameter, which has an impact on convergence time, will be discussed in Section 6.

If a node considers it has gained the slot, it keeps transmitting in the same slot in the next cycle. Thus, when all nodes have found the slot in which to transmit without collision, the network enters a beacon collision-free operation.

The pseudocode of the decentralised beacon scheduling algorithm is shown in Algorithm 1 , where $rand(1)$ denotes a uniformly random value between 0 and 1, both inclusive.

While operating in the collision-free phase, if a new node joins the network, it selects a slot among the free slots in its 2-hop neighbourhood, so no further disturbances occur. In case more than one node joins the network at the same time and selects the same slot, some of the nodes move to the previous phase until a new collision-free schedule is found.

5. Formulation as a constraint satisfaction problem solver

We model the network as an undirected graph $G = (V, E)$, with number of nodes $N = |V|$ and edges $(i, j) \in E$, with $(i \leftrightarrow j)$ denoting that nodes i and j are neighbours. Since beacon transmissions will collide at a given receiver if any of its neighbouring nodes transmit

[2] Considering a success if the slot is advertised by some neighbouring nodes as empty solves the problem of link asymmetry, as will be discussed in Section 8.

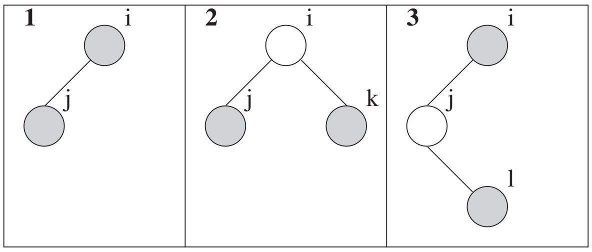


Fig. 3. Causes for node i to be dissatisfied. Filling patterns represent slot selections.

the beacon in the same slot, we are interested in finding a proper colouring of the distance-2 graph $G_2 = (V, E_2)$, with $N = |V|$ and edges $(i, j) \in E_2$, with $(i \leftrightarrow j)_2$ meaning that nodes i and j are neighbours or have a neighbour in common (1-hop and 2-hop neighbours, respectively). Then, node i successfully transmits a beacon if its selected slot is not used by any j with $(i \leftrightarrow j)_2$.

We follow the notation in [25], where the first decentralised constraint satisfaction problem solver was presented. We define N variables, one for each node in the network, and let $\vec{x} := (x_1, \ldots, x_N)$ be the vector of slots selected by all nodes with x_i being the slot selected by sensor node $i \in \{1, \ldots, N\}$. Having $\mathcal{C} = \{1, \ldots, C\}$ as the set of slots in the schedule, then the vector $\vec{x} \in \mathcal{C}^N$. We will also define a set of clauses (constraints) per edge, each clause being $\Phi_m(\vec{x}), m \in \{1, \ldots, |E|\}$. Clause $\Phi_m(\vec{x})$ evaluates to 1 if the clause is fulfilled and to 0 otherwise.

The clause for the m-th edge $(i \leftrightarrow j)$ in E is defined as a compound of three different subclauses as shown in Eq. (1).

$$\Phi_m(\vec{x}) = \begin{cases} 1 & \text{if} \\ 0 & \text{otherwise.} \end{cases} \quad a_m \wedge b_m \wedge c_m \tag{1}$$

where the three subclauses are,

$$a_m = (x_i \neq x_j). \tag{2a}$$

$$b_m = (x_j \neq x_k), \forall (k \leftrightarrow i), k : k \neq j. \tag{2b}$$

$$c_m = (x_i \neq x_l), \forall (l \leftrightarrow j), l : (l, i) \notin E. \tag{2c}$$

Eq. (2a) evaluates to 0 if the 1-hop neighbour j is using the same slot as node i (problem depicted in Fig. 3(1)). The subclause in Eq. (2b) is not fulfilled if the neighbour j shares the same slot as any other neighbour k of node i (problem shown in Fig. 3(2)). Finally, subclause in Eq. (2c) is not satisfied if node i shares the same slot as node l that is 2-hops away, with node j being the neighbour in common (issue shown in Fig. 3(3)).

The participation set of node i ($\mathcal{M}_i$) is formed by the set of clauses in which node i is at the originating side of the edge:

$$\mathcal{M}_i = \{m \equiv (i, j) : (i, j) \in E\}. \tag{3}$$

Node i is said to be satisfied if all the clauses in its participation set $\mathcal{M}_i$ are fulfilled. Therefore, we only need that node i evaluates Eq. 4 to decide whether it is satisfied.

$$\min_{m \in \mathcal{M}_i} \Phi_m(\vec{x}). \tag{4}$$

Theorem 1. *A slot selection $\vec{x}$ that satisfies $\min_i \min_{m \in \mathcal{M}_i} \Phi_m(\vec{x}) = 1, i \in \{1, \ldots, N\}$ is a proper colouring of the distance-2 graph G_2.*

Proof. Having $\Phi_m(\vec{x}) = 1, m \in \mathcal{M}_i$ for all nodes in the network corresponds to the case where there are no collisions of beacons among 1-hop neighbours (Eq. (2a)) and that no node is overhearing a collision of beacons among neighbouring nodes (Eq. (2b)). Thus, the slot selection $\vec{x}$ ensures no other node in the 2-hop neighbourhood is using the same slot as node $i, \forall i \in \{1, \ldots, N\}$. □

To evaluate Eq. (4), it is not necessary that a node is able to specifically evaluate all the different clauses in $\mathcal{M}_i$. A node just needs to know whether x_i is a satisfactory assignment, but it is not needed that it knows neither which subclause(s), nor which clause(s), is/are not fulfilled.

Theorem 2. *All clauses in $\mathcal{M}_i$ are satisfied iff node i receives a beacon from all its neighbours including a flag acknowledging the correct reception of its previously transmitted beacon.*

Proof. By receiving a collision flag in a beacon from at least one neighbour, node i knows that either a 1-hop or a 2-hop neighbour selected the same slot (either subclauses a or c in Eq. (1) are not satisfied). Alternatively, when overhearing a collision of beacons in the neighbourhood, subclause b in Eq. (1) is not satisfied. The lack of at least one beacon from a neighbour in the given schedule implies that the beacon of the target node is colliding with the beacon transmission of that neighbour, thus subclause a in Eq. (1) is not satisfied.

Thus, if no collision flag is received, no collision in the neighbourhood is overheard and all beacons from identified neighbours are received, node i knows its slot selection is not being used in its 2-hop neighbourhood and can evaluate $\min_{m\in\mathcal{M}_i}\Phi_m(\vec{x})=1$. Otherwise, it evaluates $\min_{m\in\mathcal{M}_i}\Phi_m(\vec{x})=0$. □

When node i is dissatisfied ($\min_{m\in\mathcal{M}_i}\Phi_m(\vec{x})=0$) it updates the next selection of x_i based on the probability vector $\vec{p}_i\in[0,1]^C$. Using L-ZC, $\vec{p}_i$ takes value $\gamma\in(0,1)$ for some $j\in\{1,\dots,C\}$, 0 for the slots node i is aware that are already used in the 2-hop neighbourhood and $(1-\gamma)/c^{(i)}(t)$ for some $k\in\{1,\dots,C\}$ with $k\neq j$, where $c^{(i)}(t)$ denotes the number of slots node i sees as free in schedule t. On the contrary, if the node is satisfied, the vector $\vec{p}_i$ takes value 1 for the previously selected slot and 0 for the rest. When all nodes are satisfied, the network enters in an absorbing collision-free operation state.

6. Convergence analysis

Since information of the slots used in the 2-hop neighbourhood is exchanged in beacons and nodes refrain from selecting slots seen as used in the 2-hop neighbourhood, it is not always possible to move in one cycle from having 2 dissatisfied nodes to a state in which all nodes in the network are dissatisfied. Note that, even in the case of collisions of beacons when certain information cannot be assessed, nodes may be able to obtain the slot selection in their neighbourhood via other neighbours. Therefore, convergence proofs as the one presented in [26] or the extension described in [27], in which an approach based on a *flame-front* of dissatisfied nodes is considered, are not directly applicable in this case.

To prove convergence to collision-free operation, we assume that the number of slots in the schedule is at least equal to the number of nodes in the largest 2-hop neighbourhood ($C\geq\Delta_2+1$ where Δ_2 is the maximum degree of the graph G_2). In Section 8, we will discuss the case in which there are insufficient slots in the schedule to allow for collision-free operation.

Theorem 3. *Given $C\geq\Delta_2+1$, convergence to a schedule that satisfies $\min_i\min_{m\in\mathcal{M}_i}\Phi_m(\vec{x})=1, i\in\{1,\dots,N\}$ is almost surely reached in finite time for any initial selection of slots.*

Proof. Suppose at some time the network has not reached convergence, so there are n nodes that are not satisfied, with $n\leq N$. If all dissatisfied nodes sequentially select the spare slot in their 2-hop neighbourhood while the rest remain fixed, after, at worse, N steps the network will have reached convergence. Observe that, since we consider $C\geq\Delta_2+1$, having at least one dissatisfied node in the 2-hop neighbourhood means that there is at least one slot not used by any of the nodes in its 2-hop neighbourhood. Therefore, the probability of having reached convergence in N schedules is bounded below by:

$$L:=\left(\frac{1-\gamma}{C-1}\gamma^{N-1}\right)^N=\left(\frac{1-\gamma}{C-1}\right)^N\gamma^{N^2-N}>0 \tag{5}$$

Note that, after selecting the spare slot in the 2-hop neighbourhood, subclauses a and c are satisfied, but the node can still be dissatisfied due to overhearing a collision among neighbours (subclause b). This is the reason why after selecting the spare slot, we require the node to stick to it in the subsequent steps. Observe also that we have considered the worst case $n=N$, although some of them might be satisfied. The probability that a satisfied node selects a slot not used by any of its 2-hop neighbours in the next schedule is 1 (it has previously selected it), thus, the lower bound holds.

If the previously defined sequence of events has not happened after N schedules, it has the same probability of happening in the next N schedules. We define τ as the first time at which convergence is reached. The probability of reaching convergence after Nt schedules, with $t\in\mathbb{N}$, is upper bounded by:

$$P(\tau\geq Nt)\leq(1-L)^t \tag{6}$$

With $t\to\infty$, the probability of not having found convergence is 0 since:

$$\lim_{t\to\infty}P(\tau\geq t)\leq\lim_{t\to\infty}(1-L)^t=0 \tag{7}$$

We can then use this to show the algorithm converges almost surely. Let X_n be the number of dissatisfied nodes in round n of the algorithm. Note that the algorithm converges when $X_n=0$. Also note that if $X_n=0$ then $X_m=0,\forall m\geq n$, as $X_n=0$ is an absorbing state. Define $C_n:=\{\omega\in\Omega:X_n(\omega)=0\}$. Then C_n is a non-decreasing sequence of sets and $\bigcup C_n=\{\omega\in\Omega:\lim_n X_n(\omega)=0\}$. So, $\mathbb{P}(\lim X_n=0)=\mathbb{P}(\bigcup C_n)=\lim\mathbb{P}(C_n)$. However, we have shown that $\mathbb{P}(\lim X_n=0)=1$, so $\lim\mathbb{P}(C_n)=1$. Thus, we have proved that the algorithm converges almost surely, i.e., with probability one, in finite time. □

6.1. Reducing the estimated number of steps

In this subsection we briefly consider how our estimate for convergence time could be improved. Observe that the number of steps used in the proof can be reduced by noting that nodes not belonging to the same 2-hop neighbourhood can change to the spare slot simultaneously, as there is no possibility that they mutually disturb each other. This is not possible with nodes belonging to the same 2-hop neighbourhood as selecting the same slot will still result in dissatisfaction. The probability of having a set of nodes further than 2-hops away changing to the spare slot in their own 2-hop neighbourhood is:

$$\left(\frac{1-\gamma}{C-1}\right)^\nu\gamma^{N-\nu} \tag{8}$$

where ν is the cardinality of the set of nodes that can change to the spare slot, i.e., the number of 2-hop neighbourhoods considered in that step. Observe that, ν can be different at each step as by selecting a node to change the slot, the number of 2-hop neighbourhoods may be different. However, we know that $\nu\in[\delta,\zeta]$, where δ and ζ are the minimum and maximum number of 2-hop neighbourhood divisions that can be made in a graph G, i.e., the minimum and maximum number of independent sets of the graph G_2. Since Eq. (8) has its minimum at either $\nu=\delta$ or $\nu=\zeta$, depending on the value of γ, the probability of having at least δ nodes changing to the spare slot in their 2-hop neighbourhood is lower-bounded by the product of both cases (δ and ζ nodes changing to the spare slot):

$$\left(\frac{1-\gamma}{C-1}\right)^{\delta+\zeta}\gamma^{2N-\delta-\zeta} \quad (9)$$

Then, the number of steps is reduced to at most Δ_2+1 as we sequentially pick each node in a 2-hop neighbourhood and make it, plus the rest of nodes not interfering among themselves, to select the spare slot. Therefore, the probability of reaching convergence in Δ_2+1 schedules is bounded below by:

$$L' := \left[\left(\frac{1-\gamma}{C-1}\right)^{\delta+\zeta}\gamma^{2N-\delta-\zeta}\right]^{\Delta_2+1} > 0 \quad (10)$$

Similarly, the probability of reaching convergence for the first time after $(\Delta_2+1)t$ schedules, with $t\in\mathbb{N}$, is:

$$P(\tau \geq (\Delta_2+1)t) \leq (1-L')^t \quad (11)$$

which also tends to 0 as t tends to infinity.

The lower bound L' in Eq. (10) is not always higher than the one found in Eq. (5) as it depends on the values of variables γ, N, C, δ and ζ. However, if the graph is not complete, we know that Δ_2+1 will always be smaller than N. Observe also that, in practice we expect convergence to be faster than these bounds (Eq. (6) and (11)) as we have considered very specific sequences of events.

7. Algorithm benchmarking

We have first evaluated the algorithm presented in this work in random unit disk graphs with given characteristics in order to obtain conclusions under controlled settings. In particular, we have evaluated the algorithm in different graphs with the goal of obtaining the convergence rate for finding a proper schedule assignment. These target scenarios are described and convergence rates are analysed in different conditions (by varying the γ parameter and the number of available slots in the schedule). For the evaluation, we use a custom Matlab simulator. Then, we have considered more realistic graphs to obtain the times to convergence in close-to-reality networks.

7.1. Random unit disk graphs

We are interested in analysing the convergence rate of the proposed protocol in scenarios where no initial planning of the position of nodes is performed. These scenarios are the worst cases, in the sense that no prior topology information can be inferred. For this purpose we define the scenarios of interest to be random unit disk graphs and vary the average node degree as performance depends significantly on the node degree distribution. When selecting our candidate graphs, we have discarded random graphs with large deviation in the degree distribution. Specifically, we have computed the 95%-percentile of the distribution of the node degrees of the graph G and discarded those graphs with 95%-percentile larger than $\lceil(5/4)\mathrm{av}(\deg(G))\rceil+1$, where $\mathrm{av}(\deg(G))$ denotes the average degree of the graph G. More realistic graphs are considered later in this section.

We have considered the following metrics of a graph: *(i)* the average degree of the graph G (number of 1-hop neighbours), *(ii)* the average degree of the graph G_2 (1 and 2-hop neighbours) and *(iii)* a modified count of the average C4 motif degree. C4 motifs are defined as non-induced graphs in which 4 vertexes are connected in a cycle [28]. The motif degree counting can give us more information on the interactions among 2-hop neighbouring nodes. In particular, the average C4 motif degree shows whether a large number of neighbours do have a neighbouring node in common. A large C4 motif degree indicates the case depicted in Fig. 3(3) may be more likely to occur. Common neighbours of the 1-hop neighbourhood are not able to reuse the same slots and so, the average C4 motif degree is likely to have an impact on the convergence rate. However, in our case, we are interested in the situation in which this common neighbour is not connected to the selected node. Note that, otherwise, this node will also be a neighbour of the selected node, case depicted in Fig. 3(1) and (2) and its effect is already captured by the node degree distribution. To compute this modified C4 motif degree count, we use the algorithm in [28], but adapted to count cases in which there is no edge between the node of interest and the furthest node in the cycle.

Table 1
Characteristics of the unit disk graphs selected.

deg(G) (av./std./max.)	deg(G$_2$) (av./std./max.)	deg$_{c4}$(G) (av./std.)
5/1.83/11	11.58/3.87/21	4.05/3.96
7/1.95/12	16.93/4.72/30	13.46/8.89
8/2.11/13	20.32/5.23/32	21.09/12.74
9/2.51/17	22.04/5.89/35	32.04/16.66

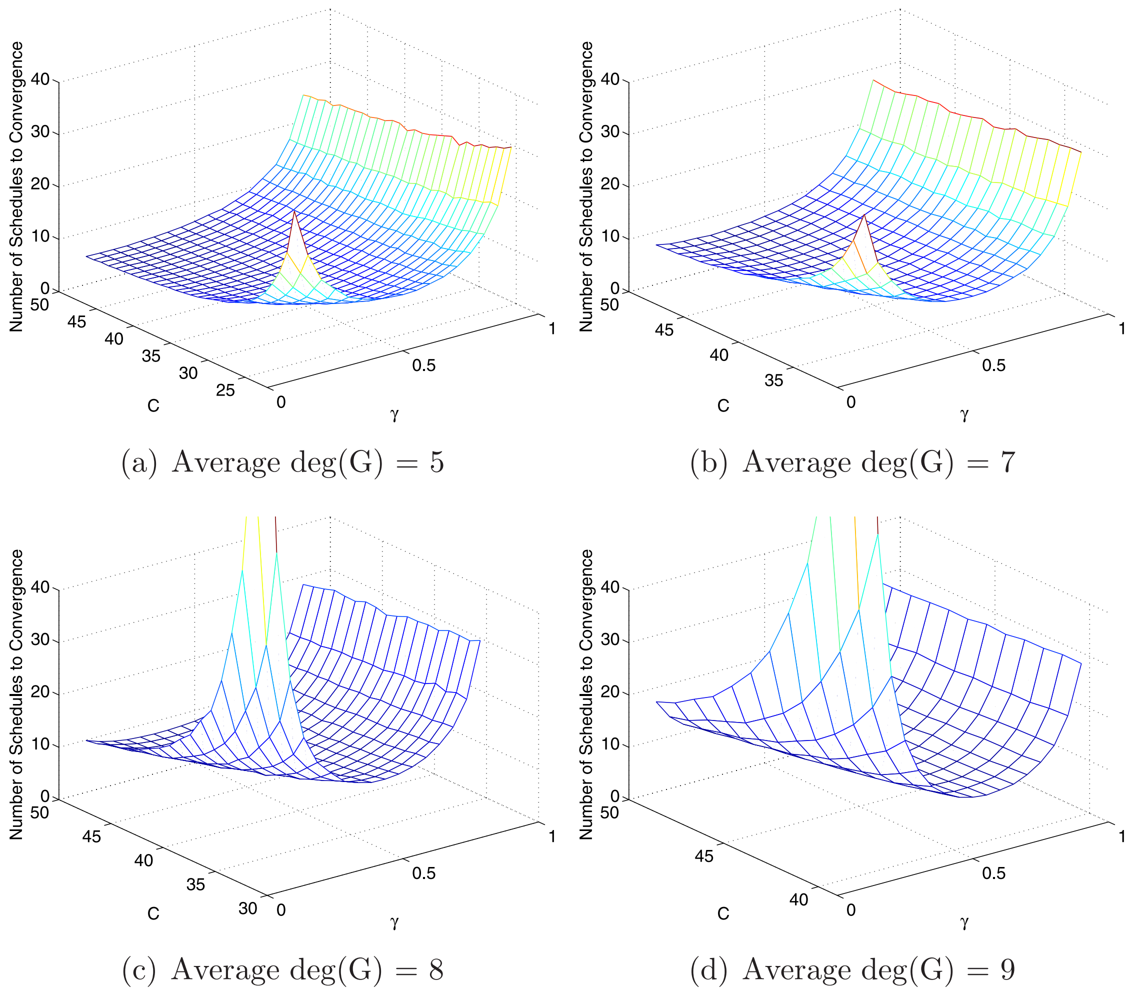
Table 1 shows the characteristics of the 3 different random graphs used for the performance evaluation (with $N = 190$). We denote by deg(G), deg(G$_2$) and deg$_{c4}$(G) the sets of node degrees of the graph G, G_2 and C4 motif degrees of G, respectively. It is important to emphasise that, once the average degree of G is fixed, there is not significant variation in these other metrics, and so we cannot use these particular graphs to determine how each parameter individually affects the convergence rate. However, we have depicted the characteristics of the graphs used for the evaluation in Table 1 to provide the maximum information of the graphs used for the evaluation.

7.2. Convergence rate for random unit disk graphs

The times to convergence for different values of C and γ in the graphs selected for evaluation are depicted in Fig. 4. Values shown are average results of 1000 simulation runs.

Observe that the selection of γ has a considerable impact on the time to convergence for certain configurations. First, when the number of slots in the schedule (C) is reduced, a small value of γ substantially increases the number of schedules needed to converge. With a small value of γ, nodes change the current selected slot after negative feedback with higher probability. Thus, when there is a small number of empty slots in the schedule, the probability for selecting the same slot as another node in a 1 and 2-hop neighbourhood increases and so does the average time to reach a satisfactory schedule assignment. Note that, this effect has a huge impact in the time to reach convergence when the number of nodes in the neighbourhood increases (Fig. 4(c) and 4(d)). In this case, an increased value of the γ parameter substantially reduces the time to convergence. However, when γ becomes large enough, the time to convergence increases, although not to the same levels as when γ is small. In fact, independently of the number of slots in the schedule, remaining on the same slot with a high probability provides an increased time to convergence as nodes keep colliding on the same slot for longer. On the other hand, when the number of slots in the schedule is high compared to the number of nodes in the 2-hop neighbourhood (there are more empty slots to select from), the time to reach convergence remains similar for moderate to small values of the γ parameter.

It is worth observing that, overall, values in the middle range of possible values of γ provide a reasonable good performance and that no substantial gain is obtained by fine tuning γ inside that range. This finding will be discussed in more detail in the following section.



(a) Average deg(G) = 5
(b) Average deg(G) = 7
(c) Average deg(G) = 8
(d) Average deg(G) = 9

Fig. 4. Number of schedules to convergence in a random unit disk graph varying C (the number of slots in the schedule) and γ (the learning parameter).

7.3. Setting the learning parameter

As we have just shown, the value of the learning parameter (γ) significantly influences the time to convergence. Small values of the learning parameter (i.e., higher probability to change the slot) combined with few slots to spare substantially increases the expected time to find a proper assignment. On the other hand, high values of γ (high probability to remain in the same slot) when a large number of slots are free, is not efficient.

Analysis to find the optimal learning parameter was already performed in [9] for the special case of coordinating unicast packet transmissions. However, since the characteristics of the proposal presented in this work substantially differ from [9], the value of the optimal γ presented in that work is not applicable here. In particular, *(i)* here we have considered a network in which not all nodes are in coverage range, thus, different conditions seen at each node can lead to a different value of γ for every node and *(ii)* nodes are considered unsatisfied if they overhear a collision of beacons (even if their slot is not actually causing a conflict).

In the scenario addressed in this work, the optimal value of γ for a target node depends on factors such as its number of 1 and 2-hop neighbouring nodes, the number of free slots in the schedule and the interconnections between neighbours. So, to derive the optimal value of γ is analytically complex and in practice limited information is available to each node. In terms of game theory, every node has 2 possible actions to choose from: *(i)* remain using the same slot and *(ii)* change to a free slot in the schedule. We consider a uniformly mixed strategy, i.e., a given action is chosen following a uniform probability function. The value of γ, considering that there are 2 possible actions to choose from, will then be equal to 0.5. We have already made the observation that, for the scenarios of interest, setting γ in the middle range of possible values gives reasonable good results and that no substantial benefits can be obtained by fine tuning inside this range. We now evaluate in detail which is the penalty of using the uniformly mixed strategy compared to the minimum convergence time obtained from simulations varying the γ parameter in different scenarios.

We show in Fig. 5, for the different graphs selected, the times to convergence obtained by setting $\gamma = 0.5$ and the minimum value obtained from Fig. 4 (average as well as 95% confidence intervals are provided). As can be observed, when the number of slots in the schedule is comparable to the number of nodes in the 2-hop neighbourhood, there is a negligible penalty and that there is discrepancy when the number of slots in the schedule substantially diverges. However, the latter penalty is no more than 1 schedule, even considering 95% confidence intervals, for the cases evaluated. As a consequence, it can be stated that for the kind of scenarios evaluated (which are common in wireless networks), setting γ to 0.5 provides a reasonable good performance without the cost involved in designing a mechanism for the nodes to be able to compute this parameter in an optimal manner.

7.3.1. Convergence rate for realistic graphs

Although considering random disk unit graphs allows us to derive conclusions based on their characteristics, to evaluate the algorithm in more realistic graphs is crucial. In this section, we

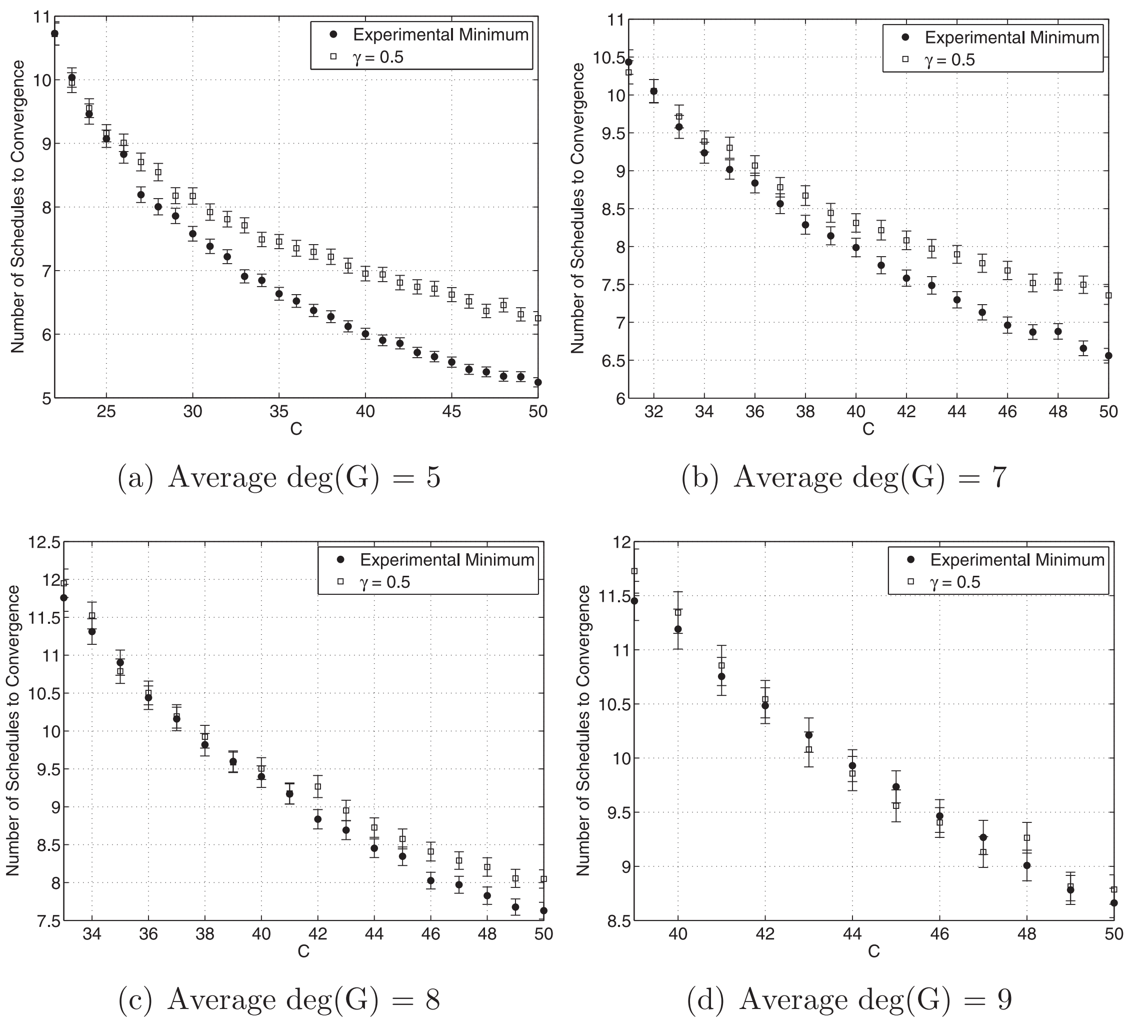


Fig. 5. Comparison among the minimum number of schedules to convergence obtained from simulations and setting $\gamma = 0.5$ while varying C (the number of slots in the schedule).

Table 2
Characteristics of the graphs drawn from *Wigle*.

N	deg(G) (av./std./max.)	deg(G_2) (av./std./max.)	deg_{c4}(G) (av./std.)
27	2.74/1.85/7	3.15/2.71/7	2.30/4.40
96	55.13/27.73/73	58.73/27.07/82	1.73/7.00

evaluate the convergence rate in two representative graphs drawn from the *Wigle* [29] database. The 27-node graph used in [30] as well as a graph built considering the location of 96 access points in a 150 m^2 area at the junction of 5th Avenue and 59th Street in Manhattan are considered (nodes are considered neighbours when located at $\leq$ 30 m distance). Observe in Table 2, that shows the characteristics of the graphs selected, how the densities of these graphs substantially differ.

Fig. 6(a) and (b) show the number of schedules to convergence for different C and $\gamma = 0.5$ for the 27 and 96-node graphs, respectively. When large enough to be shown, 95% confidence intervals are depicted. Note how the density of the graphs considered affects the range of C for which a fast convergence rate is achieved. However, it can be observed that even setting γ equal to 0.5 fast convergence to collision-free operation is obtained for reasonably large values of schedule lengths in high-density graphs.

8. Practical implications

In this section we discuss some practical implications such as non-ideal channel-condition and clock drift considerations as well as the applicability of the proposed approach to the protocols described in Section 2.

8.1. Non-ideal channel conditions

Here, we discuss the implications of noise and external interference, link asymmetries and the capture effect on the correct protocol operation.

Noise and external interference: Noise and external interference do not affect the correct protocol operation. However, they can make the convergence time of the protocol increase since erroneous receptions of beacons could make nodes incorrectly infer that a collision has occurred. Thus, nodes may change their selected slot, increasing the time to reach convergence or triggering changes in the schedule, when it would not be necessary.

Link asymmetries: Due to link asymmetries, a node can receive a beacon from a neighbour advertising its previously selected slot as empty. That occurs if the given node is able to detect its neighbouring transmissions but the same does not apply in the other direction. Since there is no conflict in this case, the node can continue using the same slot. Note that if the links are symmetric, a

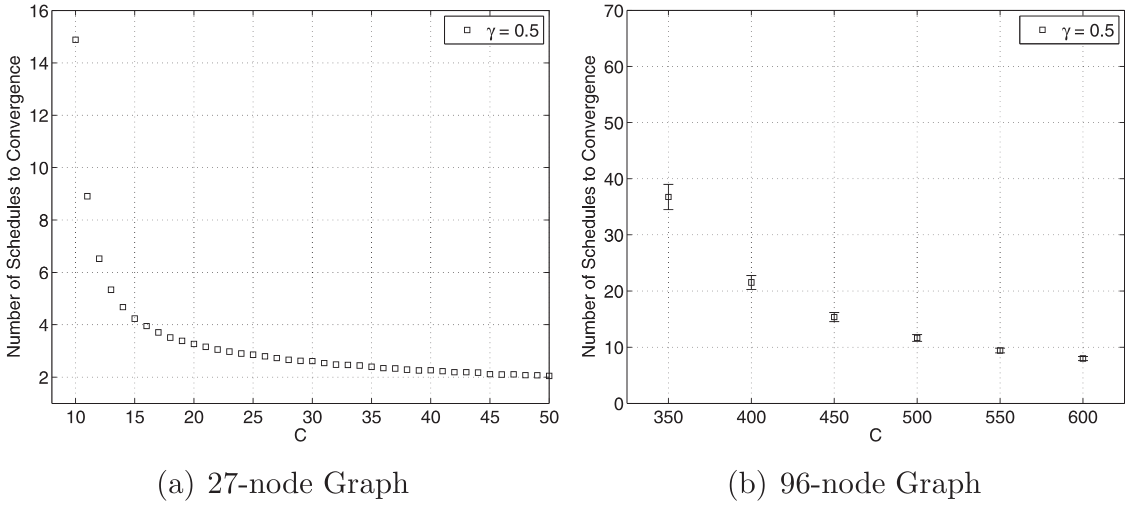


Fig. 6. Number of schedules to convergence in the graphs drawn from *Wigle* varying *C* (the number of slots in the schedule).

node may advertise a slot as empty even though a beacon was transmitted due to instantaneous channel errors. In that case we rely on the eventual correct reception and consequent advertisement of a successful slot. If another node uses the same slot while it is advertised as empty, we also rely on the eventual detection of the conflict. This case relates to the situation where the neighbouring node advertises the slot as occupied but due to another beacon reception. In that case, there is not a conflict occurring since there is no collision taking place at the receiver. Therefore, it is also safe in this case for that node to keep using the currently selected slot.

Capture effect: In case of capture, a recipient may be in the range of multiple beacons in a given slot and still be able to correctly decode one of them. Thus, it will advertise that slot as a successful reception even that there is actually a conflict. Nodes involved in the conflict would not be able to realise the problem is occurring if that receiver is the only neighbour in common or if all the common neighbouring nodes are able to capture one of the beacons sent. However, as these collisions will be successively occurring, it is likely that eventually the receiver(s) will not be able to capture any of the beacons colliding and will, therefore, flag the collision.

8.2. Clock synchronisation imperfections

One important aspect for the correct operation of the protocol relates to clock synchronisation imperfections of wireless cards. Note that beacons must be transmitted in their allocated slot and aligned to the start of the slot boundary for the correct detection at neighbouring nodes which expect the beacon reception at that specific time. However, this problem has been studied before in [31] from an experimental point of view. In particular, the authors in [31] demonstrate that a collision-free protocol in which messages are sent at specific time instants, thus, similar to the one proposed in this work, is feasible in practice even considering network card imperfections.

8.3. Insufficient number of slots in the schedule

Note that the main purpose of our approach is to schedule beacon transmissions to maintain synchronisation, allow for network discovery and reduce energy consumption. Consequently using a long schedule does not have a substantial impact on performance. In contrast, when the messages to be scheduled are data packets, and especially in WLANs, the length of the schedule plays an important role in the determination of throughput and latency. Thus, for scheduling beacons, the impact of having some empty slots in the beacon schedule is low, and so we mainly consider the case where the designer allows longer than necessary schedules.

On the other hand, in the case where there are insufficient empty slots, a node will select a busy slot in the beacon schedule. In such a case convergence cannot be achieved by any protocol. However, for the proposed protocol nodes will sporadically be able to transmit their beacons, because of random reassignment. Thus, even in this case, permanent collisions are not expected to occur.

Note that as we include information on the conditions seen in the last schedule, it can happen that, even though enough slots are available in the current schedule, nodes advertise otherwise. In this case, selecting a slot uniformly at random allows nodes to attempt transmission in the given schedule.

8.4. Applicability to current standards and networks

The proposal presented here can be applied to IEEE 802.15.4e with minor changes to the standard. That is, extending the bitmap to include information about previous beacon collisions, probabilistically transmitting a beacon in a slot and relying on information included in neighbouring beacons instead of relying on *beacon allocation notification commands* and *beacon collision notification commands*. These changes aim to make the beacon scheduling robust by enabling nodes to work in a beacon collision-free operation.

Regarding the applicability to IEEE 802.11s, one of the main problems is that we require beacons to be sent at predefined time intervals. The IEEE 802.11s can be adapted in order to make nodes transmit their beacons in the subsequent slots after beacon receptions from neighbours or by defining a time interval between consecutive beacon transmissions from neighbouring nodes, in a similar way as done in the multi-supreframe defined in IEEE 802.15.4e. All these require a change in the standard but at the benefit of achieving beacon collision-free operation.

To apply the proposed approach to receiver-initiated protocols in WSNs it requires the inclusion of the bitmap in beacons sent and then nodes may wake up at approximately the same time to receive neighbouring transmissions, see [5] for more details.

9. Final remarks

We have presented a completely decentralised and parsimonious mechanism for collision free-operation of beacon transmissions based on learning. The solution aims to solve crucial problems in current wireless mesh networks, such as those based on IEEE 802.15.4e and IEEE 802.11s standards, where beacon transmissions may successively collide making it difficult to discover neighbouring nodes and maintain synchronisation. It can also be used to efficiently support broadcast traffic in receiver-initiated WSNs.

The proposed algorithm converges almost surely in finite time and the actual time to convergence in the scenarios of interest is low, making it quite practical for mesh networks involving sporadic mobility. We have also defined how to select the learning parameter in order to: *i)* keep the time to convergence low and *ii)* maintain the protocol simplicity and low overhead.

Finally, we have considered the practical implications of deploying the presented mechanism considering non-ideal channel conditions as well as taking into account its integration in current standardisation efforts. We believe the proposed approach can be readily implemented in IEEE 802.15.4e and IEEE 802.11s, among others, with some changes in the standards.

Acknowledgment

This work has been partially supported by the Science Foundation Ireland grant 08/SRC/I1403 and 07/SK/I1216a.

References

[1] IEEE 802.15.4e, Standard for Local and Metropolitan Area Networks. Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs). Amendment 1: MAC sublayer (16 April 2012).
[2] IEEE 802.11s, Standard for Local and Metropolitan Area Networks. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. Amendment 10: Mesh Networking (10 September 2011).
[3] R. de Paz Alberola, B. Villaverde, D. Pesch, Distributed duty cycle management (DDCM) for IEEE 802.15.4 beacon-enabled wireless mesh sensor networks, in: The IEEE 8th International Conference on Mobile Adhoc and Sensor Systems (MASS), 2011, pp. 721–726.
[4] Y. Sun, O. Gurewitz, D. Johnson, RI-MAC: a receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks, in: The 6th ACM Conference on Embedded Network Sensor Systems (Sensys), 2008, pp. 1–14.
[5] C. Cano, D. Malone, B. Bellalta, J. Barceló, On the improvement of receiver-initiated MAC protocols for WSNs by applying scheduling, The IEEE 14th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013.
[6] K.-I. Hwang, S.-W. Nam, Analysis and enhancement of IEEE 802.15. 4e DSME beacon scheduling model, J. Appl. Math. 2014 (2014).
[7] J. Lee, J. Walrand, Design and Analysis of an Asynchronous Zero Collision MAC Protocol, Arxiv preprint arXiv:0806.3542(2008).
[8] J. Barcelo, B. Bellalta, C. Cano, M. Oliver, Learning-BEB: avoiding collisions in WLAN, Eunice Summer School, 2008.
[9] M. Fang, D. Malone, K. Duffy, D. Leith, Decentralised learning MACs for collision-free access in WLANs, Wireless Netw. (2012).
[10] A. Koubaa, A. Cunha, M. Alves, A time division beacon scheduling mechanism for IEEE 802.15. 4/Zigbee cluster-tree wireless sensor networks, in: The IEEE 19th Euromicro Conference on Real-Time Systems (ECRTS), 2007, pp. 125–135.
[11] J. Cho, S. An, An adaptive beacon scheduling mechanism using power control in cluster-tree WPANs, Wireless Pers. Commun. 50 (2) (2009) 143–160.
[12] R. Burda, C. Wietfeld, A distributed and autonomous beacon scheduling algorithm for IEEE 802.15. 4/ZigBee networks, in: The IEEE 4th International Conference on Mobile Adhoc and Sensor Systems, 2007, pp. 1–6.
[13] P. Muthukumaran, R. Paz, R. Spinar, D. Pesch, MeshMAC: enabling mesh networking over IEEE 802.15. 4 through distributed beacon scheduling, Ad Hoc Netw. 28 (2010) 561–575.
[14] S. Pollin, M. Ergen, S.C. Ergen, B. Bougard, I. Moerman, A. Bahai, P. Varaiya, F. Catthoor, Performance analysis of slotted carrier sense IEEE 802.15.4 medium access layer, IEEE Trans. Wireless Commun. 7 (9) (2008) 3359–3371.
[15] W. Lee, K. Hwang, Y. Jeon, S. Choi, Distributed fast beacon scheduling for mesh networks, in: The IEEE 8th International Conference on Mobile Adhoc and Sensor Systems (MASS), 2011, pp. 727–732.
[16] C.-M. Wong, C.-F. Chang, B.-H. Lee, A simple time shift scheme for beacon broadcasting based on cluster-tree IEEE 802.15.4 low-rate WPANs, Wireless Pers. Commun. 72 (4) (2013) 2837–2848.
[17] O. Iova, F. Theoleyre, M. Zou, J.-l. Lu, Efficient and reliable MAC-layer broadcast for IEEE 802.15.4 wireless sensor networks, in: Proceedings of the 7th IFIP Wireless and Mobile Networking Conference (WMNC), 2014, pp. 1–8.
[18] J. Lee, E. Rhee, Dynamic beacon collision detection/avoidance mechanism in WLAN-based mesh networks, Int. J. Digit. Content Technol. Appl. 7 (11) (2013).
[19] E.-Y. Lin, J.M. Rabaey, A. Wolisz, Power-efficient rendez-vous schemes for dense wireless sensor networks, in: Proceedings of the IEEE International Conference on Communications, 7, 2004, pp. 3769–3776.
[20] D. Kominami, M. Sugano, M. Murata, T. Hatauchi, Y. Fukuyama, Performance evaluation of intermittent receiver-driven data transmission on wireless sensor networks, in: Proceedings of the 6th International Symposium on Wireless Communication Systems., 2009, pp. 141–145.
[21] F.-J. Wu, Y.-C. Tseng, Distributed wake-up scheduling for data collection in tree-based wireless sensor networks, IEEE Commun. Lett. 13 (11) (2009) 850–852.
[22] M.-S. Pan, P.-L. Liu, Y.-P. Lin, Event data collection in ZigBee tree-based wireless sensor networks, Elsevier Comput. Netw. 73 (2014) 142–153.
[23] X. Ma, E.L. Lloyd, A distributed protocol for adaptive broadcast scheduling in packet radio networks, The 2nd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL M for Mobility), 1998.
[24] C. Zhu, M.S. Corson, A five-phase reservation protocol (FPRP) for mobile Ad Hoc networks, in: The IEEE Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)., 1, 1998, pp. 322–331.
[25] K.R. Duffy, C. Bordenave, D.J. Leith, Decentralized constraint satisfaction, IEEE/ACM Trans. Netw. 21 (4) (2013) 1298–1308.
[26] D. Leith, P. Clifford, A self-managed distributed channel selection algorithm for WLANs, in: The IEEE 4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, 2006, pp. 1–9.
[27] K. Duffy, N. O'Connell, A. Sapozhnikov, Complexity analysis of a decentralised graph colouring algorithm, Inf. Process. Lett. 107 (2) (2008) 60–63.
[28] D. Marcus, Y. Shavitt, Efficient counting of network motifs., in: ICDCS Workshops, 2010, pp. 92–98.
[29] Wireless Geographic Logging Engine "wigle", http://www.wigle.net/.
[30] A. Mishra, V. Shrivastava, D. Agrawal, S. Banerjee, S. Ganguly, Distributed channel management in uncoordinated wireless environments, in: Proceedings of the 12th Annual International Conference on Mobile Computing and Networking, 2006, pp. 170–181.
[31] L. Sanabria-Russo, F. Gringoli, J. Barcelo, B. Bellalta, Implementation and experimental evaluation of a collision-free MAC protocol for WLANs, in: Proceedings of the IEEE International Conference on Communications. (ICC 15), 2015.

Cristina Cano obtained the Telecommunications Engineering Degree at the Universitat Politecnica de Catalunya (UPC) in February 2006. Then, she received her M.Sc. (2007) and Ph.D. (2011) on Information, Communication and Audiovisual Media Technologies from the Universitat Pompeu Fabra (UPF). From July 2012 to December 2014, she was working as a research fellow at the Hamilton Institute in the National University of Ireland, Maynooth (NUIM). Part of this research group is now based in Trinity College Dublin (School of Computer Science and Statistics), where she was working from January 2015 to January 2016 as a senior research fellow. Now she is with the FUN team at Inria-Lille. Her research interests are related to coexistence of wireless heterogeneous networks, distributed scheduling, performance evaluation and stability analysis.

David Malone received B.A. (mod), M.Sc. and Ph.D. degrees in Mathematics from Trinity College Dublin. During his time as a postgraduate, he became a member of the FreeBSD development team. He is currently a SFI Stokes Lecturer at the Hamilton Institute, NUI Maynooth. His interests include mathematics of networks, network measurement, IPv6 and systems administration. He is a coauthor of O'Reilly's IPv6 Network Administration.