H-NAMe: A Hidden-Node Avoidance Mechanism for Wireless Sensor Networks

Anis Koubâa[1,2], Ricardo Severino[1], Mário Alves[1], Eduardo Tovar[1]

[1] IPP-HURRAY! Research Group, Polytechnic Institute of Porto, School of Engineering (ISEP-IPP)
Rua António Bernardino de Almeida, 431, 4200-072 Porto, Portugal

[2] Al-Imam Muhammad Ibn Saud University, Computer Science Dept., 11681 Riyadh, Saudi Arabia

{aska, rars, mjf, emt}@isep.ipp.pt

Abstract

The hidden-node problem has been shown to be a major source of Quality-of-Service (QoS) degradation in Wireless Sensor Networks (WSNs) due to factors such as the limited communication range of sensor nodes, link asymmetry and the characteristics of the physical environment. In wireless contention-based Medium Access Control protocols, if two nodes that are not visible to each other transmit to a third node that is visible to the formers, there will be a collision – usually called hidden-node or blind collision. This problem greatly affects network throughput, energy-efficiency and message transfer delays, which might be particularly dramatic in large-scale WSNs. This paper tackles the hidden-node problem in WSNs and proposes H-NAMe, a simple yet efficient distributed mechanism to overcome it. H-NAMe relies on a grouping strategy that splits each cluster of a WSN into disjoint groups of non-hidden nodes and then scales to multiple clusters via a cluster grouping strategy that guarantees no transmission interference between overlapping clusters. We also show that the H-NAMe mechanism can be easily applied to the IEEE 802.15.4/ZigBee protocols with only minor add-ons and ensuring backward compatibility with the standard specifications. We demonstrate the feasibility of H-NAMe via an experimental test-bed, showing that it increases network throughput and transmission success probability up to twice the values obtained without H-NAMe. We believe that the results in this paper will be quite useful in efficiently enabling IEEE 802.15.4/ZigBee as a WSN protocol.

1. Introduction

[1]In the last few years, wireless networking communities have been directing increasing efforts in pushing forward *anywhere and anytime* distributed computing systems. These efforts have lead to the emergence of smart device networking, including Wireless Sensor Networks (WSNs), which represent enabling infrastructures for large-scale ubiquitous and pervasive computing systems. However, a limitation for the large-scale deployment of WSNs is the relatively poor performance in terms of throughput due to the use of contention-based Medium Access Control (MAC) protocols, such as the CSMA (Carrier Sense Multiple Access) family. Such expectation is intuitively vindicated by the impact of the hidden-node problem, which is caused by hidden-node collisions.

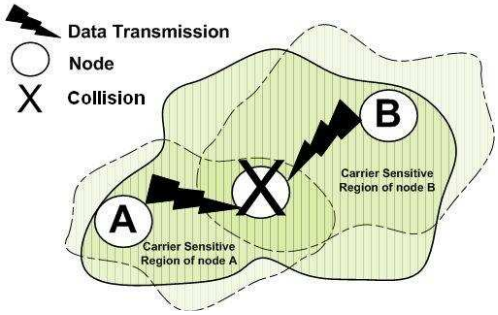

Fig. 1: A hidden-node collision

A hidden-node (or "blind") collision occurs when two nodes, which are not visible to each other (due to limited transmission range, presence of asymmetric links, presence of obstacles, etc.), communicate with a commonly visible node during a given time interval as illustrated in Fig.1. This leads to the degradation of the following three performance metrics.

1. *Throughput*, which denotes the amount of traffic successfully received by a destination node and that decreases due to additional blind collisions.
2. *Energy-efficiency* that decreases since each collision causes a new retransmission.
3. *Transfer delay*, which represents the time duration from the generation of a message until its correct reception by the destination node, and that becomes larger due to the multiple retransmissions of a collided message.

Fig. 2 presents an example obtained with our OPNET [1] simulation model [2] for the IEEE 802.15.4 protocol [3], just to highlight the negative impact of the hidden-node problem. We considered a star network spanning on a square surface (100x100 m^2) with 100 nodes, where traffic generation followed a Poisson distribution. The throughput is shown for different transmission ranges of the nodes. We vary the transmission range of the nodes by setting different receiver sensitivity levels. The degradation of the throughput performance due to hidden-node collisions is clearly noticeable in Fig. 2. This is due to the increase of

[1] This work was partially funded by FCT under the CISTER Research Unit (FCT UI 608) and by the CONET Cooperating Objects Network of Excellence.

the hidden-node collision probability when decreasing the transmission range.

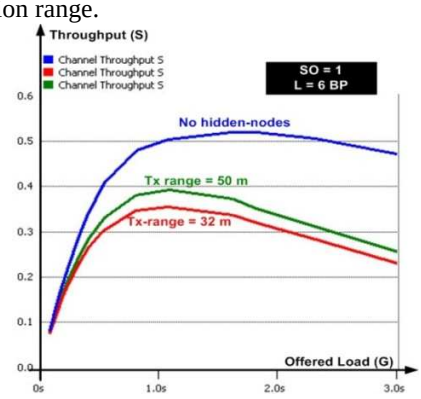

Fig. 2: Hidden-node impact on network throughput

In the literature, several mechanisms (which we briefly discuss in Section 2) have been proposed to resolve or mitigate the impact of the hidden-node problem in wireless networks. However, to our best knowledge, no effective solution to this problem in WSNs was proposed so far.

This paper proposes an efficient solution to the hidden-node problem in synchronized cluster-based WSNs. Our approach is called H-NAMe and is based on a grouping strategy that splits each cluster of a WSN into disjoint groups of non-hidden nodes. It then scales to multiple clusters via a cluster grouping strategy that guarantees no transmission interference between overlapping clusters.

The recently standardized IEEE 802.15.4/ZigBee protocol stack, which is considered as a promising candidate for WSNs (e.g. [4]), supports no hidden-node avoidance mechanism. This leads to a significant QoS degradation as already seen in Fig. 2. The resolution of this problem is of paramount importance for improving reliability, throughput and energy-efficiency. In this line, we show the integration of the H-NAMe mechanism in the IEEE 802.15.4/ZigBee protocols, requiring only minor add-ons and ensuring backward compatibility with their standard specifications. We developed an experimental test-bed and carried out a significant number of experiments showing that H-NAMe increases network throughput and transmission success probability up to 100%, against the native IEEE 802.15.4 protocol.

We believe that the integration of the H-NAMe mechanism in IEEE 802.15.4/ZigBee may be relevant in leveraging the use of these protocols in WSNs and in enriching future versions of their specifications.

Contributions of this paper. The main contributions of this paper are three-folded:

- First, we propose H-NAMe, a simple and efficient mechanism for solving the hidden-node problem in synchronized multiple cluster WSNs (Section 3).
- Second, we show how to incorporate H-NAMe in the IEEE 802.15.4/ZigBee protocols (Section 4).
- Third, we demonstrate the feasibility of the H-NAMe mechanism through an experimental test-bed and show its practical benefit (Section 5).

2. Related Work

The hidden-node problem has been shown to be a serious problem that degrades the performance of wireless networks. In [5, 6], the authors have derived a mathematical analysis based on queuing theory and have quantified the impact of the hidden-node problem on the performance of small-scale linear wireless networks. On the other hand, most of research works have focused on finding solutions for eliminating or reducing the impact of the hidden-node problem in wireless networks. Hidden-node avoidance mechanisms can be roughly categorized as follows.

- *The busy tone mechanism*.

In this approach, a node that is currently hearing an ongoing transmission sends a busy tone on a narrow band channel to its neighbors for preventing them from transmitting during channel use. This mechanism was early introduced in [5], providing a solution, called the *Busy Tone Multiple Access* (BTMA), for a star network with a base station. An extension of this mechanism for a distributed peer-to-peer network has been proposed in [8] known as Receiver-initiated Busy Tone Multiple Access (RI-BTMA) and in [9] as Dual Busy Tone Multiple Access (DBTMA).

The limitation of this mechanism is the need of a separate channel, leading to additional hardware cost and complexity, thus reducing the cost-effectiveness of WSNs.

- *RTS/CTS mechanism*

The idea of making a channel reservation around the sender and the receiver through a control-signal handshake mechanism was first proposed in [12] – SRMA (*Split-channel Reservation Multiple Access*). The Request-To-Send/Clear-To-Send (RTS/CTS) approach builds on this concept and was first introduced in the MACA protocol [13]. The channel reservation is initiated by the sender, which sends an RTS frame and waits for a CTS frame from the destination, before starting the effective transmission. Several refinements were proposed, including MACAW [14], the IEEE 802.11 (DCF) [15] and FAMA [16]. Recently, the Double Sense Multiple Access (DSMA) mechanism was proposed in [17], joining the busy tone approach with the RTS/CTS mechanism, using two time-slotted channels.

This method is particularly unsuitable for WSNs, as stated in [18], mainly due to the following reasons: (i) data frames in WSNs are typically as small as RTS/CTS frames, leading to the same collision probability; (ii) the RTS/CTS exchanges are power consuming for both the sender and the receiver and (iii) the use of RTS/CTS is only limited to unicast transmissions and does not extend to broadcasts. In addition, it may lead to extra throughput degradation due to the *exposed-node* problem [13].

- *Carrier Sense Tuning*

The idea consists in tuning the receiver sensitivity threshold of the transceiver, which represents the minimum energy level that indicates channel activity, to have extended radio coverage. Higher receiver sensitivities

enable a node to detect the transmissions of nodes farther away, thus leading it to defer its transmission to avoid overlapping. Many works analyzed the impact of carrier sensing on the system performance. This technique was analyzed in [19] to study the effects of carrier sensing range on the performance of the IEEE 802.11 MAC protocol. A similar study was conducted in [20], where the authors derived expressions of the number of hidden nodes that may affect a given sender and the corresponding probability of collision. More recently, in [21] the authors carry out a thorough study to find an optimal carrier sensing threshold, given multiple network topologies.

The limitation of carrier sense tuning is that it assumes homogenous radio channels, whereas in reality, hidden-node situations can arise from obstacles and asymmetric links. In addition, it is not possible to indefinitely increase the carrier sense range due to physical limitations.

- *Node Grouping*

Node grouping consists in grouping nodes according to their hidden-node relationship, such that each group contains nodes that are "visible" (bidirectional connectivity) to each other. Then, these groups are scheduled to communicate in non-overlapping time periods to avoid hidden-node collisions. Such a grouping strategy is particularly suitable for star-based topologies with one base station. In that direction, a grouping strategy was recently introduced in [22] to solve the hidden-node problem in IEEE 802.15.4/ZigBee star networks (formed by the ZigBee Coordinator – ZC – and several nodes in its coverage). In [22], the grouping strategy assumes that the ZC can distinguish a hidden-node collision from a normal collision based on the time when the collision occurs. Thus, when the ZC detects a hidden-node collision, it starts the hidden-node information collection process, by triggering a polling mechanism. At the end of the polling process, all nodes report their hidden-node information to the ZC, which executes a group assignment algorithm based on the hidden-node relationship reported by the nodes. The algorithm used in shown to have a complexity of $O(N^2)$. After assigning each node to a group, the ZC allocates to each group a certain time duration inside the superframe, in which slotted CSMA/CA is used as MAC protocol. The grouping process is then repeated each time the ZC detects a hidden-node collision.

In this paper, we propose a really efficient, practical and scalable approach to the case of cluster-based WSNs. We also show how to integrate our approach in IEEE 802.15.4/ZigBee protocols with only minor add-ons and backward compatibility.

Our work differs from [22] in many aspects, overcoming important limitations. First, H-NAMe requires no hidden-node detection since it relies on a *proactive* approach rather than a *reactive* approach to the hidden-node problem. Hence, our grouping strategy is node-initiated. Second, we reduce the complexity of the group join process. The grouping process in [22] is based on polling all the nodes in the coverage of ZC each time a hidden-node collision occurs with a group assignment complexity of $O(N^2)$ in each grouping process, where N is the number of nodes. This results in network inaccessibility time and energy consumption during the polling process. In our approach, for each group assignment, only the requesting node and its neighbors will be subject to the *group join* procedure and not all the nodes of the cluster, resulting in a simpler and more energy-efficient ($\sim O(N)$) mechanism. Third, we show how to scale our mechanism to multiple cluster networks. Finally, we demonstrate the feasibility of our proposal through a real test-bed, whereas the [22] relies on simulation. This is quite relevant, because we believe an eventual implementation of [22] would not be straightforward, since it requires a mechanism for detecting and interpreting collisions, implying a non-negligible change to the IEEE 802.15.4 Physical Layer.

3. The H-NAMe mechanism

3.1. System model

We consider a multiple cluster wireless network and we assume that in each cluster there is at least one node with bi-directional radio connectivity with all the other cluster nodes (Fig. 3). We denote this node as Cluster-Head (CH). At least the CH must support routing capabilities, for guaranteeing total interconnectivity between cluster nodes.

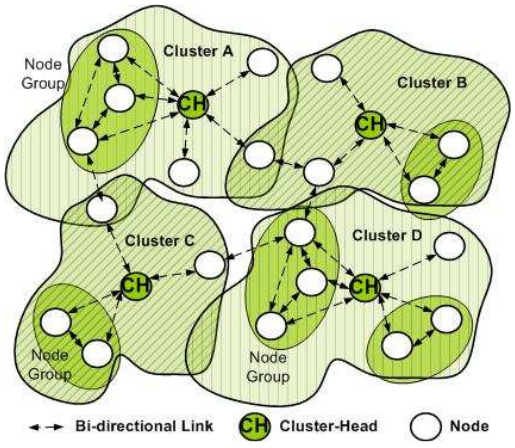

Fig. 3 : Network model

Nodes are assumed to contend for medium access during a Contention Access Period (CAP), using a contention-based MAC (e.g. CSMA family). A synchronization service must exist to assure synchronization services to all network nodes, either in a centralized (e.g. GPS, RF pulse) or distributed fashion (e.g. IEEE 802.11 TSF, ZigBee). We also assume that there is interconnectivity between all network clusters (e.g. mesh or tree-like topology). Note that although our current aim is to use the H-NAMe mechanism in the IEEE 802.15.4/ZigBee protocols, the system model is generic enough to enable the application of H-NAMe to other wireless communication protocols (e.g. IEEE 802.11).

In what follows, we start by proposing the H-NAMe intra-cluster node grouping strategy (Section 3.2) and then,

in Section 3.3, a strategy to ensure the scalability to multiple cluster networks.

3.2. Intra-cluster grouping

Initially, all nodes in each cluster share the same CAP, thus are prone to hidden-node collisions. The H-NAMe mechanism subdivides each cluster into node groups (where all nodes have bi-directional connectivity) and assigns a different time window to each group during the CAP. The set of time windows assigned to node groups transmissions is defined as Group Access Period (GAP), and must be smaller or equal to the CAP. In this way, nodes belonging to groups can transmit without the risk of hidden-node collisions.

For the intra-cluster grouping mechanism, we start by assuming that there is no interference with adjacent clusters, since that might also instigate hidden-node collisions.

The H-NAMe intra-cluster grouping strategy comprises four steps, presented hereafter and illustrated in Fig. 4.

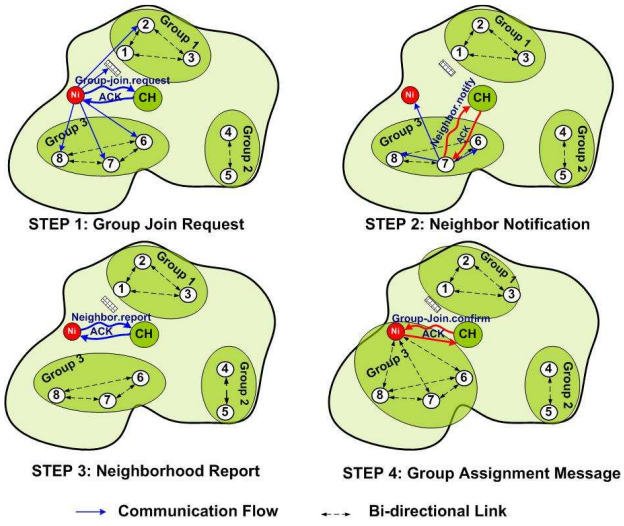

Fig. 4: Intra-cluster grouping mechanism

A message sequence diagram is presented in Fig. 5.

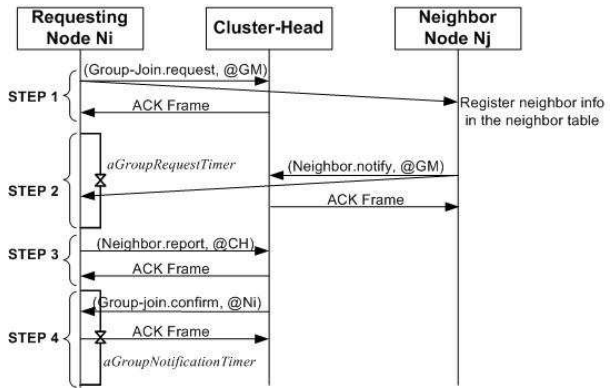

Fig. 5: Intra-cluster grouping message sequence chart

Step 1 - Group Join Request

Let us consider a node N_i that wants to avoid hidden-node collisions. Node N_i sends a *Group-join.request* message to its cluster-head CH, using a specific broadcast address referred to as *group management address* $@_{GM}$ in the destination address field. $@_{GM}$ is defined as an *intra-cluster broadcast address*, which must be acknowledged by the cluster-head (in contrast to the typical broadcast address). Obviously, the acknowledgment message (ACK) will be received by all cluster nodes, since the cluster-head is assumed to have bi-directional links with all of them.

Such an acknowledged broadcast transmission ensures that the broadcasted message is correctly received by all the neighbors of the broadcasting node (recalling that we assume no inter-cluster interference). In fact, if any collision occurs inside the cluster during the transmission of the broadcast message, then the cluster-head CH will certainly be affected by this collision since it is in direct visibility with all nodes in its cluster. If no collision occurs, then the broadcast message will be correctly received by all nodes and acknowledged by the cluster-head.

Hence, since the *Group-join.request* message is sent using the group management address $@_{GM}$, CH sends back an ACK frame to N_i notifying it of the correct reception of the group join request.

On the other side, all cluster nodes in the transmission range of N_i (thus received the *Group-join.request* message) and that already belong to a group, check if they have N_i already registered as a neighbor node in their *Neighbor Table*. We assume that the Neighbor Table is created and updated by each node during network set-up and run-time phases. The Neighbor Table stores the addresses of neighbor nodes and the link symmetry information, which specifies if the link with a corresponding neighbor is bi-directional or not. If a node hears the *Group-join.request* message and does not belong to any group (it is transmitting in the CAP, thus not in the GAP), then it simply ignores the message. On the other hand, if a node N_j is already in a group and hears the join message, then it records the information about N_i in its Neighbor Table, if it is not registered yet, and will update the link symmetry with direction $N_i \rightarrow N_j$.

Step Status. At the end of this step, each node in the transmission range of N_i knows that node N_i is asking for joining a group and registers the neighborhood information of N_i. This only ensures a link direction from N_i to this set of nodes. The link symmetry verification is the purpose of the next step.

Step 2 - Neighbor Notification

After receiving the ACK frame of its *Group-join.request* message, node N_i triggers the *aGroupRequestTimer* timer, during which it waits for neighbor notification messages from its neighbors that heard its request to join a group and that already belong to a group. Choosing the optimal duration of this timer is out of the scope of this paper, but it must be large enough to permit all neighbors to send their notification.

During that time period, all nodes that have heard the join request and that already belong to a group must initiate a *Neighbor.notify* message to inform node N_i that they have heard its request. One option is that a node N_j directly sends the *Neighbor.notify* message to node N_i with an acknowledgement request. The drawback of this alternative is that node N_j cannot know when its *Neighbor.notify* message fails to reach N_i (i.e. ACK frame not received), whether the lost message is due a collision or to the non-visibility of N_i. No clear decision can be taken in that case. A better alternative is that node N_j sends the *Neighbor.notify* message using the group management address $@_{GM}$ in the destination address field. As previously mentioned, the correct reception of the *Neighbor.notify* message by the cluster-head CH followed by an ACK frame means that this message is not corrupted by any collision and is correctly received by all nodes in the transmission range of N_j. Particularly, node N_i will correctly receive the neighbor notification message if it is reachable from node N_j; otherwise, the link between N_i and N_j is unidirectional (direction $N_i \rightarrow N_j$). If N_i receives the *Neighbor.notify* message from N_j, then it updates its Neighbor Table by adding as a new entry the information on N_j with *Link Symmetry* set to bi-directional ($N_i \leftrightarrow N_j$), if this information has not been recorded yet. If N_j has already been registered as a neighbor node, N_i must be sure to set the *Link Symmetry* property to bi-directional. This procedure is executed by all nodes responding to the *Group-join.request* message during the timer period *aGroupRequestTimer*.

Step Status. At the end of this step, the requesting node N_i will have the information on all bi-directional neighbors that have already been assigned to groups. Since N_i does not know the number of nodes in each group, it cannot decide alone which group it will join. The group assignment is the purpose of the next steps.

Step 3 – Neighbor Information Report

The cluster-head CH is assumed to be the central node that manages all the groups in its cluster. Thus, CH has a full knowledge of the groups and their organization. For that reason, after the expiration of the *aGroupRequestTimer* timer, node N_i sends the *Neighbor.report* message, which contains the list of its neighbor nodes (that have been collected during the previous step), to its cluster-head CH (using the CH address $@_{CH}$ as a destination address). The CH must send back an ACK frame to confirm the reception. Then, node N_i waits for a notification from CH that decides whether N_i will be assigned to a group or not. CH must send the group assignment notification before the expiration of a time period equal to *aGroupNotificationTimer*. If the timer expires, node N_i concludes that its group join request has failed and may retry to join a group later.

Step Status. At the end of this step, N_i will be waiting for the group assignment confirmation message from CH, which tries to assign N_i to a group based on its neighbor information report and the organization of the groups in its cluster. The group assignment procedure and notification is presented in the next step.

Step 4 - Group Assignment Procedure

The cluster-head CH maintains the list of existing groups. After receiving from node N_i the *Neighbor.report* message containing the list of its bi-directional neighbors, CH starts the group assignment procedure to potentially assign N_i to a given group, according to its neighborhood list and available resources. In each cluster, the number of groups must be kept as low as possible in order to reduce the number of state information that needs to be managed by the CH.

We impose that the number of groups inside each cluster must not exceed *aMaxGroupNumber*, which should be equal to six, by default (the reader is referred to [22] for further intuition). The group assignment algorithm is presented in Fig. 6:

Group Assignment Algorithm

```
1   int aMaxGroupNumber; // maximum number of groups
2                        in a cluster
3   Type Group;
4   Group G;        // list of all groups G[1]..G[aMaxGroupNumber]
5   |G[i]| = number of elements in group G[i]
6   Type Neighbor_List;      // {Np .. Nq)= Neighbor List of
7                               the requesting Node N
8   int Count [|G[i]|] = {0, 0, .., 0};  // Number of nodes in Neighbor
9                                           List that belongs to the group G[i]
10  int grp_nbr;   // the current number of groups managed by CH
11  // group_index function returns the group index of the node NL[i]
12  function int group_index(Neighbor_List NL, int i)
13  //the group assignment function.
14  int group_assign (Neighbor_List NL, Group G, int grp_nbr) {
15     int res = 0;
16     int index = 0;
17     while ((res = = 0) and (index < |NL|)) {
18        if (++Count[group_index (NL, index)] = =
19                                 |G[group_index (NL, index++])|)
20           res = group_index (NL, --index); break;
21     }
22     if (res = = 0) {    //that means that no group is found
23        if (grp_nbr = = aMaxGroupNumber) return (res)
24           else return (++grp_nbr);
25     }
26    else return (res);
27  }
```

Fig. 6. Group assignment algorithm

Upon reception of the *Neighbor.report* message, the cluster-head CH checks the neighbor list of the requesting node N_i. If there is a group whose (all) nodes are neighbors of node N_i, then N_i will be associated to that group. The cluster-head runs the following algorithm (as in Fig. 6). For each neighbor node N_j in the list, the cluster-head CH increments *Count* [*group_index* (N_j)], which denotes the number of neighbor nodes of N_i that belong to the group of the currently selected neighbor N_j. Note that *group_index*

(Nj) denotes the index of the group of node N_j. If this number is equal to the actual number of nodes of the latter group, it results that all nodes in this group are neighbors of node N_i. Thus, N_i can be assigned to this group since it is visible to all its nodes.

If the list of neighbors is run through without satisfying such a condition, the cluster-head CH will create a new group for N_i if the number of groups is lower than *aMaxGroupNumber*; otherwise, the *Group-join.request* message of N_i will be considered as failed. So it must transmit during the CAP (not in the GAP), and may retry a new group join request later.

At the end of the group assignment process, CH sends a *Group-join.notify* message to node N_i to notify it about the result of its group join request.

If the requesting node is assigned a group, then it will be allowed to contend for medium access during the time period reserved for the group, which is called *Group Access Period* (*GAP*). This information on the time period allocated to the group is retrieved in the subsequent frames sent by the CH.

Importantly, the complexity of the algorithm (Fig. 6) for assigning a group to a node depends on the number of neighbors of this node. In any case, it is smaller than $O(N)$, where N is the number of nodes in the cluster, thus has significantly lower complexity than the $O(N^2)$ complexity of the algorithm for group assignment proposed in [22]. Moreover, in that proposal each new node that enters the network is unaware of the existing groups and will cause a hidden-node collision, after which the groups are re-constructed. In our mechanism, a node is not allowed to transmit during the time period allocated to groups (only being able to communicate during the CAP) until it is assigned to a given group.

Group load-balancing: Note that the algorithm presented in Fig. 6 stops when a first group of non-hidden nodes is found for the requesting node. However, a requesting node can be in the range of two different groups, i.e. all nodes in two separate groups are visible to the requesting node. In this case, one possible criterion is to insert the requesting node into the group with the smallest number of nodes, for maintaining load-balancing between the different groups. For that purpose, the algorithm should go through all the elements of the neighbor list and determine the list of groups that satisfy the condition in lines 18 and 19 of the algorithm (Fig. 6). In this case, if more than one group satisfies this condition, N_i will be inserted in the group with the smallest number of nodes.

Bandwidth allocation: The time-duration of each group in the GAP can be tuned by the cluster-head to improve the mechanism efficiency. This can be done according to different strategies, namely: (i) evenly for all the node groups; (ii) proportionally to the number of nodes in each group; (iii) proportionally to each group's traffic requirements. How to perform this assignment is not tackled in this paper.

3.3. Scaling H-NAMe to multiple-cluster networks

Solving the hidden-node problem in multiple-cluster networks involves greater complexity due to inter-cluster interference. The assumption that there is no interference from other clusters made before is no longer valid. Hence, even if non-hidden node groups are formed inside all clusters, there is no guarantee that hidden-node collisions will not occur, since groups in one cluster are unaware of groups in adjacent clusters.

Obviously, the best strategy for completely avoiding the inter-cluster hidden-node problem is to reserve an exclusive time window for each cluster. However, this strategy is definitely not adequate for large-scale sensor networks, where the number of clusters/groups is significantly high.

Our approach consists in defining another level of grouping by creating distinct groups of clusters, whose nodes are allowed to communicate during the same time window. Therefore, each cluster group will be assigned a portion of time, during which each cluster in the cluster group will manage its own Group Access Period (GAP), according to the intra-cluster mechanism presented in Section 3.2.

The cluster grouping concept is illustrated in Fig. 3. As shown, clusters A and B have overlapping radio coverage, which can lead to inter-cluster interference and thus to hidden-node collisions. For this reason, they will be assigned to different cluster groups that are active in different time windows. The same applies for cluster pairs (C, D), (A, C) and (B, D). Therefore, our cluster grouping mechanism forms two cluster groups: Group 1, which comprises clusters A and D, and Group 2 containing clusters B and C.

The challenge is on finding the optimal cluster grouping strategy that ensures the minimum number of cluster groups. We define a cluster group as a set of clusters whose nodes are allowed to transmit at the same time without interference.

Cluster grouping and time window scheduling strategies were proposed and effectively implemented and validated in [23], for engineering ZigBee cluster-tree WSNs. A more detailed description of the cluster grouping mechanism can be found in [24][24]. We propose a grouping criterion and a graph coloring algorithm for an efficient scheduling of the cluster groups activity.

4. H-NAMe in IEEE 802.15.4/ZigBee

In this section, we explain how to instantiate the H-NAMe mechanism to the IEEE 802.15.4 protocol, namely addressing beacon-enabled cluster-tree networks. This topology is scalable and enables energy-efficient (dynamically adaptable duty-cycles per cluster) and real-time communications [27]. In addition, the cluster-tree topology fits into the H-NAMe network model.

4.1 IEEE 8021.5.4/ZigBee overview

The joint efforts of the IEEE 802.15.4 task group [25] and the ZigBee Alliance [26] have ended up with the specification of a standard protocol stack for Low-Rate Wireless Personal Area Networks (LR-WPANs), enabling low-cost and low-power wireless communications.

The IEEE 802.15.4 MAC protocol supports two operational modes that may be selected by the ZigBee Coordinator (ZC), which is the master node that identifies and manages the whole WPAN: (i) the non beacon-enabled mode, in which the MAC is simply ruled by non-slotted CSMA/CA; and (ii) the beacon-enabled mode, in which beacons are periodically sent by the ZC for synchronization and network management purposes.

In the beacon-enabled mode, the ZC defines a superframe structure (Fig. 7), which is constructed based on the Beacon Interval (*BI*), which defines the time between two consecutive beacon frames, and on the Superframe Duration (*SD*), which defines the active portion in the *BI*, and is divided into 16 equally-sized time slots, during which frame transmissions are allowed. Optionally, an inactive period is defined if *BI* > *SD*. During the inactive period (if it exists), all nodes may enter in a sleep mode (to save energy). *BI* and *SD* are determined by two parameters, the Beacon Order (*BO*) and the Superframe Order (*SO*), respectively, as follows:

$$\left.\begin{aligned} BI &= aBaseSuperframeDuration \cdot 2^{BO} \\ SD &= aBaseSuperframeDuration \cdot 2^{SO} \end{aligned}\right\} for\ \ 0 \le SO \le BO \le 14 \qquad (1)$$

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

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

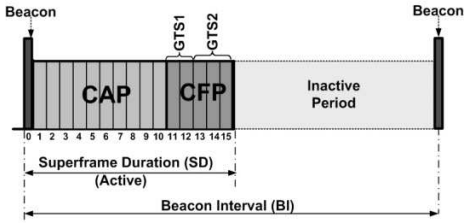

Fig. 7. Superframe structure

It can be easily observed in Fig. 7 that low duty-cycles can be configured by setting small values of the superframe order (*SO*) as compared to the beacon order (*BO*), resulting in longer sleep (inactive) periods.

ZigBee defines network and application layer services on top of the IEEE 802.15.4 protocol. The cluster-tree topology, in which all nodes are organized in a parent-child relationship, imposes the beacon-enabled mode. A simple and deterministic tree routing mechanism is used.

A ZigBee network is composed of three device types: (i) the ZigBee Coordinator (ZC), which identifies the network and provides synchronization services through the transmission of beacon frames containing the identification of the PAN and other relevant information; (iii) the ZigBee Router (ZR), which has the same functionalities as the ZC with the exception that it does not create its own PAN - a ZR must be associated to the ZC or to another ZR, providing local synchronization to its cluster (child) nodes via beacon frame transmissions; and (iii) the ZigBee End-Device (ZED), which does not have any coordination functionalities and is associated to the ZC or to a ZR.

4.2. Integrating H-NAMe in IEEE 802.15.4

Basically, the idea is that each node group (resulting from the H-NAMe mechanism) will be allocated a time window in each superframe duration. The idea is to use part of the CAP for the Group Access Period (GAP), as illustrated in Fig. 8. Note that a minimum duration of 440 symbols must be guaranteed for the CAP in each superframe [3].

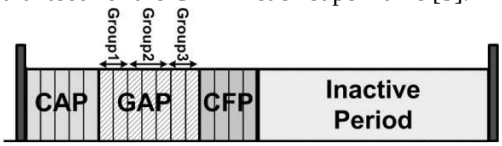

Fig. 8. CAP, GAP and CFP in the Superframe

In our intra-cluster grouping strategy, a node that has been assigned a group will track the beacon frame for information related to the time window allocated to its group, and will contend for medium access during that period with the other nodes of the same group. We propose the *GAP Specification* field in Fig. 9 to be embedded in the beacon frame (such a specification is missing in [22]).

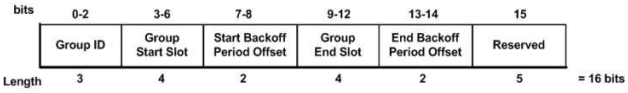

Fig. 9. GAP specification field of a beacon frame

The GAP is specified by the *Group ID* field that identifies the node group. Up to 8 groups per cluster can be defined. The time window in the superframe is specified by a given number of Backoff Periods (BP). A practical problem is that the number of a backoff period in a superframe may be quite large for high superframe orders (up to 16 time slots * 2^{16} BP/time slot), which requires a huge amount of bits in the field to express the starting BP and the final BP for each group. The objective is to maintain as low overhead as possible for the specification of a given group. For that purpose, a group is characterized by its *start time slot* and *end time slot* (between 0 and 15) and the corresponding *backoff period offsets*. The start and end offsets for the time duration of a group is computed as follows:

$$Relative\ \ Offset = (\text{Start/End})\ \text{Backoff Period Offset} * 2^{SO}$$

The choice of a *Backoff Period Offset* sub-field encoded in two bits is argued by the fact that the minimum number

of backoff periods in a time slot is equal to 3 for (SO = 0). Hence, for SO > 0, each time slot will be divided into three parts to which the start/end instant of a given group access period should be synchronized.

This GAP implementation approach only requires two bytes of overhead per group. The maximum number of groups depends on the SO values, since lower superframe orders cannot support much overhead in the beacon frame due to short superframe durations. Also, it allows a flexible and dynamic allocation of the groups, since all nodes continuously update their information about their group start and end times when receiving a beacon frame, at the beginning of each superframe.

5. Experimental Evaluation

5.1. Implementation approach

We have implemented the H-NAMe mechanism in nesC/TinyOS [28], over the Open-ZB implementation [29] of the IEEE 802.15.4/ZigBee protocols, to demonstrate its feasibility and efficiency using commercial-off-the-shelf (COTS) technologies.

For that purpose, we carried out a thorough experimental analysis to understand the impact of the H-NAMe mechanism on the network performance, namely in terms of *network throughput* (S) and *probability of successful transmissions* (Ps), for different *offered loads* (G), in one cluster with a star-based topology. Both metrics have been also used to evaluate the performance of the Slotted CSMA/CA MAC protocol [30]. The network throughput (S) represents the fraction of traffic correctly received normalized to the overall capacity of the network (250 kbps). The success probability (Ps) reflects the degree of reliability achieved by the network for successful transmissions. This metric is computed as the throughput S divided by G, representing the amount of traffic sent from the application layer to the MAC sub-layer, also normalized to the overall network capacity.

To ensure the reliability of the measurement process, some issues had to be considered, namely guaranteeing that the IEEE 802.15.4 physical channel was free from interference from IEEE 802.11 networks, which operate at the same frequency range. We have experimentally observed that despite the distance to the nearest IEEE 802.11 access point being over 10 m, it definitely impact on the performance measurements. The channel was often sensed as busy (during the Clear Channel Assessment (CCA) procedure) due to IEEE 802.11 transmissions. Hence, we chose an IEEE 802.15.4 channel outside the IEEE 802.11 frequency spectrum (Channel 26) to perform the experimental evaluation. Channel integrity was ensured using a spectrum analyzer. In addition, another aspect that was considered was the choice of the SO value to be used in our experiments. To have a clearer idea on the impact of the hidden-node phenomenon independently from other parameters, we have chosen a superframe order sufficiently high (SO = 8) to avoid the collisions related to the CCA deference problem encountered for low SO, in the slotted CSMA-CA mechanism, as presented in [30]. The CCA deference problem occurs when it is not possible for a frame to be transmitted in the remaining space of the superframe and its transmission must be deferred to the next one. For low SO and due to the lower superframe duration, it is more probable that this deference occurs (in more nodes), resulting in multiple collisions at the beginning of the next superframe. The reason is that, after the deference, the slotted CSMA-CA protocol does not perform another backoff procedure (only two CCAs).

5.2. Test-bed scenario

The experimental test-bed consisted of 18 MICAz motes [31] (featuring an Atmel ATmega128L 8-bit microcontroller with 128 kB of in-system programmable memory) scattered in three groups hidden from each other, a ZC and a protocol analyzer Chipcon CC2420 [32], capturing the traffic for processing and analysis (Fig. 10).

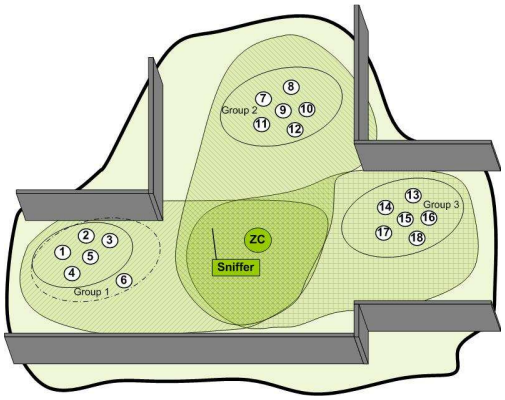

Fig. 10. Experimental testbed

The protocol analyzer generates a log file containing all the received packets and the corresponding timestamps, enabling to retrieve all the necessary data embedded in the packets payload, using a parser application we developed.

The 18 nodes have been programmed to generate traffic at the application layer with preset inter-arrival times. A similar approach has previously been used in [30] for evaluating the performance of the CSMA-CA protocol. The three node groups were placed at ground level near walls, in order to reinforce the hidden-node effect (Fig.10).

To ensure that nodes in different groups were in fact hidden, a simple test was carried out. A MICAz mote was programmed to continuously perform the clear channel assessment procedure, toggling a led when energy was detected in the channel. By placing this mote at different spots while a group of nodes was transmitting, we were able to identify an area to place a new node group so that they would be hidden from the other groups. This procedure was repeated for each group, in a way that nodes were divided evenly by the 3 groups (6 nodes/group).

5.3. Experimental results

Fig. 11 presents the GAP created by the H-NAMe mechanism.

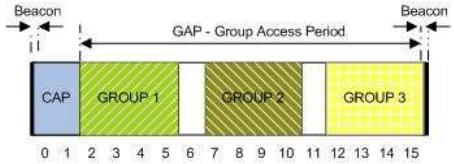

Fig. 11. Groups allocation in the superframe

Each node group was assigned with four time slots for transmission, which represents a theoretical duration of 983.04 ms per group (*SO* = 8). This allocation was made according to the principle of equal group access duration for an equal number of nodes per group.

5.3.1 The node group-join procedure

Fig. 12 illustrates a packet capture of a group join requested by a node.

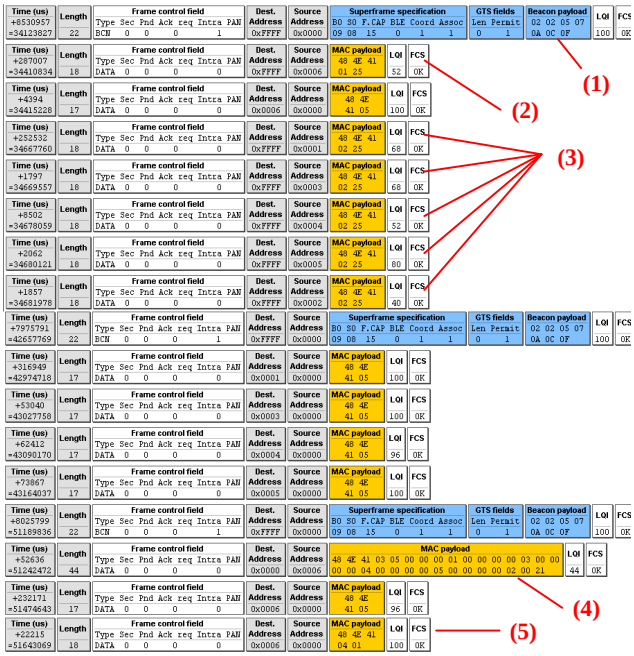

Fig. 12. Packet analyzer capture of a group join

In this example, a node with short address 0x0006 (see Fig. 10) requested to join a group. Notice the beacon payload featuring the GAP specification of the groups already formed (labeled (1) in Fig. 12).

The node initiated the process by sending a *Group-join.request* message to the ZC (label (2)) and receiving an acknowledgement. Then, all the other nodes in its transmission range replied with a *Neighbor.notify* message (label (3)). When the requesting node receives these messages, it knows that it shares a bi-directional link with its neighbors. As soon as the timer for receiving *Neighbor.notify* messages expires, the requesting node sends a *Neighbor.report* message to the ZC identifying its neighbors (label (4)). The ZC runs the H-NAMe intra-cluster grouping algorithm to assign a group to that node and sends a *Group-join.confirm* message, notifying the node of which group to join (label (5)). The node, now assigned to Group 1, can transmit during the GAP portion reserved for Group 1 (see Fig. 11).

5.3.2. H-NAMe performance evaluation

The performance evaluation of the H-NAMe mechanism has been carried out using *BO* = *SO* = 8 (100% duty cycle), with a constant frame size of 904 bits. Several runs were performed (one for each packet inter-arrival time), to evaluate the network performance at different offered loads (*G*). Fig. 13 presents the throughput (*S*) and the success probability (*Ps*) obtained from three experimental scenarios: a network with hidden-nodes without using the H-NAMe mechanism (triangle markers curve); the previous network using the H-NAMe mechanism (circle markers curve) and a network without hidden-nodes (square markers curve). The depicted average values for the throughput and probability of success were computed with a 95% confidence interval for a sample size of 3000 packets at each offered load. The respective variance is displayed at each sample point by a vertical bar in black. From these results, we can observe that even at low offered loads H-NAMe leads to a considerable performance improvement. For instance, for an offered load (*G*) of 30%, the success probability (*Ps*) using H-NAMe is roughly 50% greater than without H-NAMe.

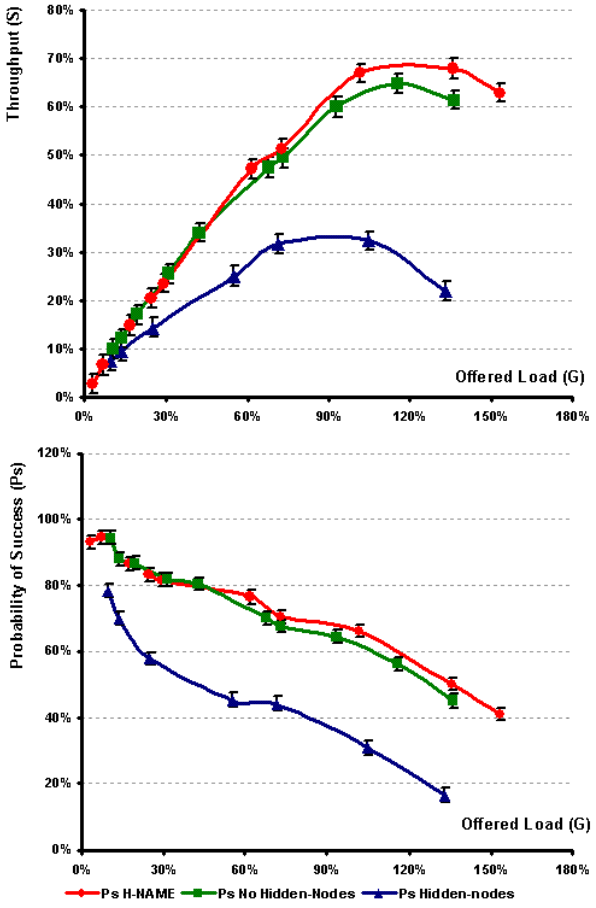

Fig. 13. Experimental performance results

Considering higher loads, it is clear that the H-NAMe doubled the throughput of the conventional network with hidden-nodes. At 90% of offered load (*G*), the throughput

of the network using H-NAMe reached 67% and is increasing, while without using H-NAMe a saturation throughput of 32% is achieved, representing an improvement of more than 100%.

Moreover, it is possible to observe that for high offered loads, the H-NAMe mechanism has actually up to 5% better throughput performance than that of a network without hidden-nodes (all nodes with bi-directional connectivity in a star topology). This results from the lower probability of collisions with H-NAMe since at most 6 nodes (one group) contend for the medium at a given time (GAP) instead of 18 nodes in the network without H-NAMe intra-cluster grouping.

In this experimental scenario, there were no packets retransmitted (due to collisions). However, if we consider one retransmission for each lost packet, the increase in the number of transmissions would be significant in the case of the network without H-NAMe, thus leading to a much higher energy loss, even at low offered loads. Notice that for G = 30%, Ps is around 50% when H-NAMe is not used, meaning that half of the packets transmitted did not reach their destination.

In conclusion, it can be noticed that the H-NAMe mechanism presents a significant improvement of the network performance in terms of throughput and success probability, at the small cost of some additional overhead to setup the different groups in the networks.

6. Concluding remarks

In this paper, we have provided a solution to a real fundamental problem in Wireless Sensor Networks (WSNs) that use contention-based medium access control (MAC) – the hidden-node problem.

We have proposed a simple yet very effective mechanism – H-NAMe – that eliminates hidden-node collisions in synchronized multiple cluster WSNs, leading to improved network throughput, energy-efficiency and message transfer delays. H-NAMe follows a proactive approach (avoids hidden-node collisions before occurring) for achieving interference-free node and cluster groups.

We have also showed how H-NAMe can easily be applied to the IEEE 802.15.4/ZigBee protocols, which are prominent candidates for WSN applications. Finally, we have implemented, tested, validated and demonstrated the feasibility and effectiveness of the H-NAMe mechanism in a real scenario, reaching a network performance improvement at the order of 100%.

References

[1] OPNET Tech., “http://www.opnet.com”, 2006.

[2] P. Jurčík, A. Koubâa, “The IEEE 802.15.4 OPNET Simulation Model: Reference Guide v2.0”, IPP-HURRAY Technical Report, HURRAY-TR-070509, May 2007.

[3] IEEE-TG15.4, "Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs),", IEEE standard for Information Technology, 2003.

[4] A. Cunha, “On the use of IEEE 802.15.4/ZigBee as federating communication protocols for Wireless Sensor Networks”, HURRAY-TR-070902, MSc Thesis, 2007.

[5] S. Ray, D. Starobinski, and J. B. Carruthers, "Performance of Wireless Networks with Hidden Nodes: A Queuing-Theoretic Analysis " Computer Communications, vol. 28, 2005.

[6] S. Ray, J. Carruthers, and D. Starobinski, "Evaluation of the Masked Node Problem in Ad-Hoc Wireless LANs," IEEE Transactions on Mobile Computing, vol. 4, 2005.

[7] F. A. Tobagi and L. Kleinrock, “Packet Switching in Radio Channels: Part II - The Hidden Terminal Problem in Carrier Sense Multiple-Access and the Busy-Tone Solution,” IEEE Transactions on Communication, vol. 23, pp. 1417-1433, 1975.

[8] C.S. Wu and V. O. K. Li, ”Receiver-initiated busy-tone multiple access in packet radio networks," in Proceedings of the ACM workshop on Frontiers in computer communications technology, Stowe, Vermont, United States, 1987.

[9] Z. J. Haas and J. Deng, "Dual busy tone multiple access (DBTMA)--A multiple access control scheme for ad hoc networks," IEEE Transactions on Communications, vol. 50, pp. 975 - 985, 2002.

[10] A. Chandra, V. Gummalla, and J. O. Limb, “Wireless collision detect (WCD): multiple access with receiver initiated feedback and carrier detect signal,” in IEEE ICC, pp. 397-401, 2000.

[11] Baowei Ji, “Asynchronous wireless collision detection with acknowledgement for wireless mesh networks”, in Proceedings of the IEEE Vehicular Technology Conference, Vol. 2, pp. 700- 704, September 2005

[12] F.A. Tobagi and L. Kleinrock, “Packet switching in radio channels: Part III – polling and (dynamic) split channel reservation multiple access”, IEEE Transactions on Computers 24(7), pp. 832–845, August 1976.

[13] P. Karn, "MACA - A New Channel Access Method for Packet Radio," in Proceedings of the ARRL/CRRL Amateur Radio 9th Computer Networking Conference, 1990.

[14] V. Bharghavan, A. Demers, S. Shenker and L. Zhang, “MACAW: A media access protocol for wireless LAN’s”, in: Proceedings of ACM SIGCOMM, London, UK, pp. 212–225, August 1994.

[15] ISO/IEC IEEE-802-11, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," IEEE standard for Information Technology, 1999.

[16] C.L. Fullmer and J.J. Garcia-Luna-Aceves, “Solutions to hidden terminal problems in wireless networks”, in: Proceedings of ACM SIGCOMM, Cannes, France, September 1997.

[17] Y. Yang, F. Huang, X. Ge, X. Zhang, X. Gu, M. Guizani, H. Chen, “Double sense multiple access for wireless ad-hoc networks”, in The International Journal of Computer and Telecomunications Networking, V. 51, Issue 14, 2007.

[18] K. Xu, M. Gerla, and S. Bae, “How effective is the IEEE 802.11 RTS/CTS handshake in ad hoc networks?,” in Proceeding of GLOBECOM’02, 2002, vol. 1, pp. 72–76

[19] J. Deng, B. Liang, and P. K. Varshney, "Tuning the carrier sensing range of IEEE 802.11 MAC," GLOBECOM - IEEE Global Telecommunications Conference, vol. 5, pp. 2987-2991, 2004.

[20] S. Xu and T. Saadawi, "Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks?," IEEE Communications Magazine, vol. 39, pp. 130-137, 2001.

[21] H. Zhai and Y. Fang, "Physical carrier sensing and spatial reuse in multirate and multihop wireless ad hoc networks," Proc. IEEE INFOCOM, April 2006.

[22] L. Hwang, "Grouping Strategy for Solving Hidden Node Problem in IEEE 802.15.4 LR-WPAN," in 1st International Conference on Wireless Internet (WICON'05). Budapest (Hungary): IEEE, 2005.

[23] A. Koubaa, A. Cunha, M. Alves, “A Time Division Beacon Scheduling Mechanism for IEEE 802.15.4/ZigBee Cluster-Tree Wireless Sensor Networks”, in Euromicro Conference on Real-Time Systems (ECRTS 2007), Pisa(Italy), July 2007.

[24] R. Severino, “On the use of IEEE 802.15.4/ZigBee for Time-Sensitive Wireless Sensor Network Applications”, MSc Thesis, Polytechnic Institute of Porto, School of Engineering, October 2008.

[25] IEEE 802.15.4 task group, “http://www.ieee802.org/15/pub/TG4b.html”, 2006.

[26] ZigBee-Alliance, “ZigBee specification," http://www.ZigBee.org/ , 2006.

[27] A. Koubâa, M. Alves, and E. Tovar, “Modeling and Worst-Case Dimensioning of Cluster-Tree Wireless Sensor Networks”, 27th IEEE Real-time Systems Symposium (RTSS’06), Rio de Janeiro, Brazil, December 2006, pp. 412-421, IEEE Computer Society.

[28] TinyOS, http://www.tinyos.net, 2007.

[29] An open-source toolset for the IEEE 802.15.4/ZigBee protocol stack, “www.open-zb.net”, 2006.

[30] A. Koubaa, M. Alves, E. Tovar, “A Comprehensive Simulation Study of Slotted CSMA/CA for IEEE 802.15.4 Wireless Sensor Networks”, In IEEE WFCS 2006, Torino (Italy), June 2006.

[31] MICAz/MICA2 mote datasheets, “http://www.xbow.com”

[32] Chipcon, "Chipcon Packet Sniffer for IEEE 802.15.4," 2006.