Congestion Reduction using a MAC Scheduling Mechanism in a Wireless Sensor Network

Nancy El Rachkidy, Alexandre Guitton

Congestion Reduction using a MAC Scheduling Mechanism in a Wireless Sensor Network

Nancy El Rachkidy[(1,2)], Alexandre Guitton[(1,2)]
(1) Clermont Université, Université Blaise Pascal, LIMOS, BP 10448, F-63000 Clermont-Ferrand, France
(2) CNRS, UMR 6158, LIMOS, F-63173 Aubière, France
Emails: {nancy,guitton}@sancy.univ-bpclermont.fr

***Abstract*—In many synchronized MAC protocols for wireless sensor networks such as IEEE 802.15.4 in beacon-enabled mode, periods where all the nodes are active alternate with periods where all the nodes are inactive. This approach is used in order to save the energy of nodes as they are powered by small batteries. However, having all nodes active simultaneously can yield to congestion, which increases the packet loss rate and the delay. In this paper, we propose a new MAC scheduling mechanism that distributes the activities of nodes into several periods, thus reducing the number of active nodes during each period. The scheduling is based on the routing information provided by the network layer. We propose an heuristic to compute this schedule, and we derive a protocol with limited overhead. We compare their performance with the performance of IEEE 802.15.4 where all nodes are active simultaneously, as well as with the optimal solution computed using an integer linear program. The simulation results show that our heuristic can greatly improve both packet loss rate and delay in a large variety of scenarios without increasing the energy consumption.**

***Index Terms*—Wireless Sensor Networks, scheduling, MAC protocols.**

I. Introduction

Nowadays, wireless sensor networks (WSNs) are used for many monitoring applications of industrial [1] or environmental sites [2] [3]. These networks are composed of small sensor nodes able to sense the environment and able to inform a central node (called sink) about the evolution of the monitored phenomenon. These nodes are powered by batteries and communicate with each other in order to form a multi-hop wireless network.

One of the challenges of protocols for WSNs is to reduce congestion. For instance, in a monitoring application where all nodes of the network send periodic traffic to a specific node of the network, called sink, congestion may arise in the whole network due to the activity of all nodes. This congestion causes an increase in the end-to-end delay and packet loss. The effects of congestion can be reduced by using efficient routing protocols and robust medium access control protocols. In this paper, we focus on a medium access control (MAC) schedule in order to further reduce the congestion (and thus increase the network performance) without impacting the energy consumption.

One of the main approaches used nowadays in order to save energy is based on deactivating the radio module of nodes periodically. In other words, the radio module of nodes is active for a given period of time (called activity period) and is inactive for another period of time (called inactivity period). To do so, the nodes have to be synchronized, which means that they have to share a common time. In most cases, the activity periods of nodes overlap completely: at a given time, either nodes are all active or they are all inactive. An example of this approach is the beacon-enabled mode of the IEEE 802.15.4 standard [4]. This standard defines the physical and the medium access control layers in a wireless personal area network, and is widely used to connect low-cost sensors and actuators.

However, this approach is not efficient when dealing with relatively high traffic load (with respect to the duty cycle) caused by traffic originating from many nodes. Indeed, as the nodes are all active during the same period, they all attempt to transmit during the same period, which increases congestion in the network and produces collisions. Thus, the packet loss rate and the end-to-end delay are both high in such a scenario [5].

This paper aims to reduce the congestion in the network (see [6] for a survey on congestion reduction for WSNs). Our protocol activates groups of nodes at different periods. For instance, one half of the nodes could be scheduled for activation first, and the other half later. In this case, the congestion experienced by each half of nodes would be low. However, this trivial scheduling would not work in most cases, as it would be impossible for nodes of the first half to communicate with nodes of the other half. This paper tackles a variation of this scheduling.

The remainder of this paper is organized as follows. In Section II, we discuss the work related to our proposition. In Section III, we describe in detail the problem, an optimal solution, our heuristic and our protocol. Simulation results are presented in Section IV. Finally, Section V concludes the paper.

II. State of the art

Existing MAC protocols for WSNs can be classified into synchronous and asynchronous protocols [7], [8].

Manuscript received June 3rd, 2014; revised August 25th, 2014.
Corresponding author email: nancy@sancy.univ-bpclermont.fr.
doi:10.12720/jcm.v.n.p-p

In synchronous MAC protocols, nodes share a common vision of time, obtained through synchronization. Then, they exchange some information to coordinate the periods of activity and inactivity (for instance, in order to reduce the time spent in idle listening). In asynchronous MAC protocols, nodes do not share a common vision of time. To ensure that a sender and a receiver can communicate, asynchronous MAC protocols use several techniques. For instance, the sender can send a large preamble before its data [9], and the receiver can wake up periodically and determine whether a preamble is currently being transmitted or not. Another approach is to send a beacon frame when a node wakes up, such that each node that receives this frame knows that it can communicate with the sender of the beacon frame. Generally, asynchronous MAC protocols yield to large end-to-end delays as a sender and a receiver first have to meet before being able to communicate.

In this paper, we focus on synchronous MAC protocols as they compute smaller delays in general. Subsection II-A gives a brief discussion on some existing synchronous MAC protocols not based on a schedule. Subsection II-B describes synchronous, schedule-based MAC protocols. In each subsection, we show the difference between these related works and our approach.

A. Synchronous MAC protocols without schedule

In the following, we describe existing synchronous MAC protocols that are not based on a schedule.

1) ZigBee and IEEE 802.15.4 standards in beacon-enabled mode: The IEEE 802.15.4 standard [4] defines the physical layer and the medium access control sublayer of a low-power wireless personal area network. It operates in two modes: the beacon-enabled mode during which periodic beacon frames are transmitted to synchronize nodes according to a superframe structure, and the non beacon-enabled mode.

In the non beacon-enabled mode, nodes are not synchronized. When a reduced function device has data to send, it wakes up and sends the data by using a channel access mechanism called unslotted carrier sense multiple access with collision avoidance (CSMA/CA). The full function devices have to be active all the time, as they ignore when reduced function devices will send data. Thus, the non beacon-enabled mode is less energy efficient than the beacon-enabled mode for full function devices.

In the beacon-enabled mode, nodes are synchronized. The activity cycle, called the superframe in the standard, is delimited by two consecutive beacon frames, and is composed of two periods: the active and the inactive periods. The active period is divided into a mandatory contention access period (CAP) and an optional contention free period (CFP). The channel access mechanism used during the CAP is called slotted CSMA/CA.

The ZigBee standard [10] defines the upper layers of a network stack based on IEEE 802.15.4. It uses the ad-hoc on demande distance vector (AODV) routing protocol [11] and allows multi-hop communications (that are out of the scope of IEEE 802.15.4).

In our approach, we modify the activation periods of IEEE 802.15.4: rather than activating all nodes simultaneously, we activate them in groups. We rely on the same synchronization mechanism as ZigBee/IEEE 802.15.4.

2) S-MAC (Sensor MAC) protocol: The main goal of S-MAC [12] is to reduce the energy consumption, while supporting scalability and avoiding collisions. More specifically, S-MAC tries to reduce energy consumption caused by idle listening, collisions, overhearing and control overhead. S-MAC consists of the following three major components.

- Periodic listen and sleep: for low data rate, it is not necessary to keep nodes listening all the time; using S-MAC, nodes are able to switch to sleep mode.
- Collision and overhearing avoidance: S-MAC avoids collisions by using RTS/CTS (request to send / clear to send) control packets. S-MAC also tries to limit overhearing by letting interfering nodes go to sleep briefly after hearing an RTS or a CTS packet for another destination.
- Message passing: S-MAC is able to fragment long messages into small fragments, and transmit them in burst.

While S-MAC saves energy by changing the medium access mechanism and introducing RTS and CTS control frames, the goal of our approach is to reduce congestion by deactivating several nodes for extended periods of time (and not only when an RTS or a CTS is overheard). S-MAC (as well as other protocols that optimize the channel access) can be used in addition to our approach in order to further reduce the congestion during the activity periods (of a limited number of nodes).

B. Synchronous MAC protocols with schedule

In the following, we describe existing synchronous MAC protocols that are based on a schedule.

1) TAS-MAC (Traffic-Adaptive Synchronous MAC) protocol: TAS-MAC [13] is a high throughput, low delay MAC protocol with low power consumption. It achieves high throughput by using a TDMA mechanism with a traffic-adaptive allocation mechanism. It reduces the end-to-end delay by notifying all nodes on active routes about the incoming traffic in advance. These nodes then claim time slots for data transmission and can forward packets through multiple hops in one activity cycle. The intended traffic-adaptive feature is achieved by splitting traffic notification and data transmission scheduling.

While TAS-MAC allocates time slots depending on the traffic (which is supposed to be known in advance), our approach does not make assumption about the traffic, and it is evaluated with traffic produced by each node (which is a worst-case scenario for TAS-MAC). Also, our approach is not a pure TDMA approach (unlike TAS-MAC), but can be used with CSMA/CA during the activity period (which is the scenario we evaluated, as described later in Subsection IV-A). Another difference

is that our approach is able to activate only a part of the route from a source to the sink (even though this situation degrades the performance of the network).

2) Schedule-based multi-channel MAC protocol: In [14], the authors proposed a schedule-based multi-channel MAC protocol for WSNs. Each receiving node selects a time slot in order to be able to receive from a given sender. The time slot selection is realized as follows. A node avoids to select slots that are already selected by others in its interference range. To minimize the conflicts during the time slot selection, the authors proposed to split the neighboring nodes into different groups, where nodes of a group may only select the slots allocated to this group. This protocol thus reduces congestion in the neighborhood of nodes.

The main differences between this protocol and our approach are the following. First, we aim to activate most of the nodes on the path from source to sink in order to reduce the end-to-end delay, while the authors of [14] ensure that each node has a time slot with each neighbor (which introduces a delay when the node has to wait for this time slot). The additional delay experienced by packets in their approach grows linearly with the number of nodes in a given path, and their approach suffers from large end-to-end delays. Second, our approach decorrelates slot selection and channel access: our approach can even benefit from the advantages of a CSMA/CA mechanism (which is the scenario we adopted for our simulations).

3) Crankshaft protocol: In [15], the authors proposed the Crankshaft protocol. This protocol divides time into superframes that contain two types of slots: unicast and broadcast. Each superframe starts with all the unicast slots, followed by the broadcast sots. During a broadcast slot, all nodes are active in order to listen for an incoming frame. A node having a broadcast frame to send contends with all the nodes to send that frame. Each node listens for one unicast slot (determined by its MAC address) during every superframe. During the unicast slot, a neighbouring node may send a frame to that node if it gains access to the medium. Crankshaft is based on an acknowledgment mechanism for unicast frames. If the sender does not receive an acknowledgment, the protocol tries to resend the frame three times in subsequent superframes. However, in order to reduce congestion caused by the retransmission, the node only retries the transmission of a frame with a probability of 70%.

The main difference of our approach compared to Crankshaft is that we consider a global schedule during which any medium access can be used, rather than allocating slots to nodes. Our scheduling is also based on the routes for the frames.

4) GinMAC protocol: GinMAC [16] is based on TDMA three types of slots: basic slots, additional slots, and unused slots. Each superframe contains a number of basic slots computed such that each node can forward one frame to the sink. The additional slots are used to improve the transmission reliability. The unused slots are purely used to reduce the duty cycle of nodes. These slots are of a fixed size and used in an exclusive manner: a slot used by one node cannot be re-used by other nodes in the network. GinMAC does not scale for a network with many nodes. GinMAC implements temporal and spatial transmission diversity. Indeed, it is possible to duplicate the basic schedule m times within the same GinMAC frame. Nodes in the network are then able to join $m+1$ virtual topologies. During a frame transmission, node sends a copy of the frame in each of the $m+1$ topologies. The concurrent topologies are selected by respecting the constraint that no links in common are used.

The main difference of our approach compared to GinMAC is that once the network is deployed our approach uses the same topology in order to provide reliable network performance. GinMAC on the other hand uses several virtual topologies in order to deliver data to the sink, which provides routing diversity.

5) DESYNC-TDMA: In [17], the authors introduces a desynchronized procedure into the synchronized TDMA schedule, and proposed DESYNC-TDMA, which can achieve self-organization. This protocol is able to provide high throughput and collision-free transmission under high loads. DESYNC-TDMA provides fairness and predictable message latencies. It self-adjusts to accommodate the new nodes or to recapture the unused slots.

However, DESYNC-TDMA also has some limitations and may not be suitable for all types of traffic. The main limitation is the time required to re-organize the slots when a node joins or leaves the networks.

III. MAC SCHEDULING MECHANISM

Unlike most synchronized MAC protocols, where all nodes are active simultaneously (such as ZigBee, see schedule 1 of Figure 1), our proposal consists in scheduling node activation periods so that some nodes are inactive while others are active (see schedules 2 and 3 of Figure 1). In this way, we aim at reducing the congestion caused when too many nodes are active simultaneously, and thus at reducing both packet loss rate and delay.

Our assumptions are the following. First, we assume that all nodes are synchronized, and we do not take into account the cost of the synchronization in our comparisons (as the synchronization cost is the same for all the approaches we compare). Second, we assume that the duty cycle of nodes is fixed: each node is active during the same fraction of time, and thus each approach has the same energy cost. Indeed, in WSNs, the energy spent to transmit, to receive or to listen is similar, while the energy spent when the radio module is inactive is orders of magnitudes smaller than when the radio module is active. Third, we assume that there is a sink that collects data from all the nodes, which is always active. Fourth, we assume that the routing protocol is known, and that routes do not change frequently (see Subsection III-C for a discussion on this aspect).

In Subsection III-A, we consider the simple case where nodes are divided into two groups: when nodes of one

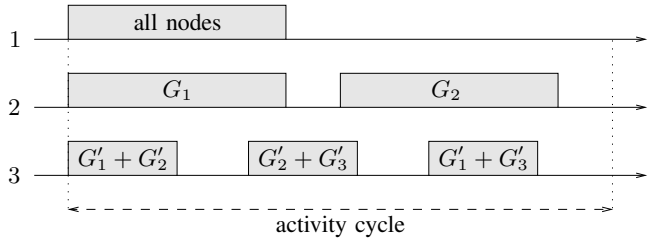


Figure 1. Three schedules of nodes with a duty cycle of $\alpha = 0.4$ (which represents 40% of the activity cycle). On schedule 1, representing ZigBee standard, all nodes are active simultaneously. On schedule 2, nodes are divided into two groups G_1 and G_2, but communication is impossible between groups. On schedule 3, nodes are divided into three groups G'_1, G'_2 and G'_3, and communication is possible between groups during some periods.

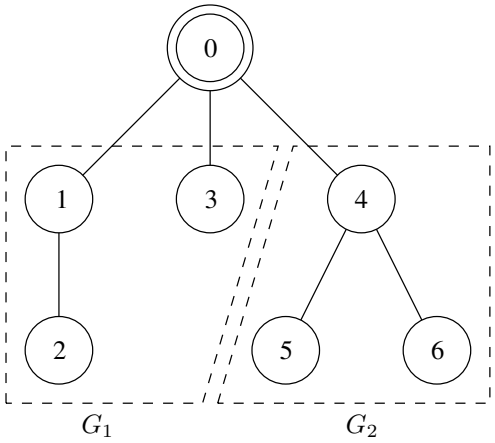


Figure 2. An example of two groups on a topology.

group are active, the nodes of the other group are inactive (see schedule 2 of Figure 1). With this approach, it is not possible for nodes of one group to communicate with nodes of the other group, which limits the possibilities of the scheduling mechanism. In Subsection III-B, we consider a more general case where nodes are divided into three groups or more. At any time, either all nodes are inactive, or all groups of nodes are active except one group (see schedule 3 of Figure 1). With this approach, it is possible for nodes of any group G'_i to communicate with nodes of any other group G'_j, although not necessarily immediately. In Subsection III-C, we describe a protocol that implements this MAC scheduling mechanism.

A. Schedule for two groups

In this subsection, we divide nodes into two groups. When the nodes of one group are active, the nodes of the other group are inactive. Thus, it is not possible for nodes of one group to communicate with nodes of the other group. Consequently, to distribute nodes into these two groups, it is essential to know all the communication paths from nodes to the sink, which form a tree rooted at the sink. The main task for our scheduling mechanism is to distribute nodes into these two groups, such that (i) all the descendants of a given child of the sink are in the same group, and (ii) the number of nodes in each group is similar. The first constraint ensures that any node can send data to the corresponding child of the sink, and in turn to the sink itself (as the sink is always active). The second constraint aims at reducing the congestion within each group.

Figure 2 shows an example of two groups G_1 and G_2 for a small topology, where the sink is node 0. When nodes of G_1 are active, all nodes of G_2 are inactive, and conversely. Note that the sink is always active. It can also be noticed that the number of nodes in each group is the same in this example.

However, distributing nodes into two groups with these constraints is an NP-complete problem. Indeed, the partition problem, which is known to be NP-complete [18] can be reduced to our problem. Recall that the partition problem aims at partitioning a set S into two subsets S_1 and S_2 such that the difference between the sum of the elements of S_1 and the sum of the elements of S_2 is minimized.

We decided to adapt a well-known greedy approach from the partition problem to our problem of distributing nodes evenly into two groups. The heuristic first computes, for each child $child$ of the sink, how many nodes (denoted by $size(child)$) are in the subtree of $child$. Then, the heuristic considers the values $size(child)$ one by one by decreasing order, and adds all the nodes of the subtree of $child$ to the group having the least number of nodes. This simple heuristic gives a 4/3-approximation of the optimal solution.

It can already be noticed that the computation of the heuristic can be done locally (and quickly) at the sink, provided that $size(child)$ is known for each child $child$ of the sink. Computing $size(child)$ for each node $child$ can be done by flooding the whole tree, which requires sending only $2(n-1)$ messages, n being the number of nodes in the network.

B. Schedule for three groups (or more)

In this subsection, we divide nodes into three groups (or more). Note that this case $m \geq 3$ is structurally different from the case $m = 2$. At any time, either all nodes are inactive, or all groups of nodes are active except for one group. This approach is shown on schedule 3 of Figure 1. With this approach, a node of a group G_i can always communicate with its next hop, even if this next hop is in another group G_j, although the communication might have to be delayed when nodes of G_j are inactive. Since each group of nodes is active during $m-1$ periods (where $m \geq 3$ is the number of groups), the activity for each period lasts for $\alpha \cdot c/(m-1)$ (c being the duration of the activity cycle and α the duty cycle).

A path from a node to the sink is said to be broken k times if there are k links (u_i, v_i) on this path such that u_i is not in the same group as v_i. When a path is broken because of a link (u_i, v_i), packets reaching u_i might have to wait for v_i to become active, thus introducing additional delay. The goal of our scheduling mechanism is thus to reduce the number of broken paths. Notice that the sink, being always active, is considered to be in all groups.

Figure 3 shows an example of three groups G'_1, G'_2 and G'_3 for a small topology. At a given time, either all groups are inactive, or two (out of three) are active. For instance, if groups G'_1 and G'_2 are active, all nodes of the the path from node 2 to node 0 are active. However, when groups G'_2 and G'_3 are active, node 2 has to wait before sending packets to node 1. Thus, the path from node 2 to node 0 is broken once. The number of nodes in each group is the same in this example.

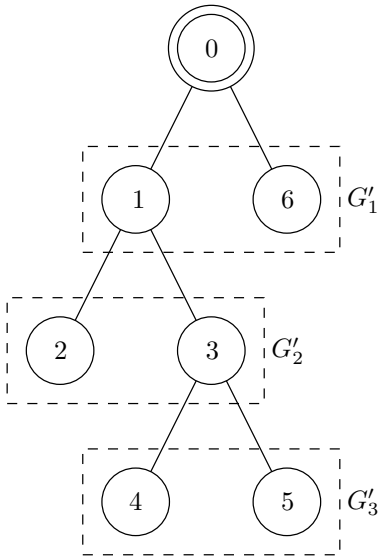


Figure 3. An example of three groups on a topology.

1) Optimal solution: In this subsection, we model the problem of distributing nodes into $m \geq 3$ groups as an integer linear program (ILP). This ILP takes as input a set of nodes N, a sink $sink \in N$, an array $father$ representing for each node the next hop towards the sink (with $father[sink] = sink$), the number of groups m, and the minimum number of nodes per group min. The objective of the ILP is to map each node to a group, while breaking the minimum number of paths. Note that paths that are broken k times are counted as k paths broken.

Our ILP uses an array of binary variables $group[x, g]$ which indicates whether node x is in group g or not, an array of binary variables $sameGroup[x, g]$ which indicates whether x and $father[x]$ are both in group g or not, an array of binary variables $brokenLink[x]$ which indicates whether the link $(x, father[x])$ is broken or not, and finally an array of binary integers $brokenPath[x]$ which indicates the number of times a path from x to $sink$ is broken.

The constraints of our ILP are given on Table I. Constraint (1) states that each node x has to be in exactly one group. Constraint (2) states that each group has a minimum size. Constraints (3), (4), (5) and (6) are used to model the fact that the link $(x, father[x])$ is broken or not. The link is broken if x and $father[x]$ are not in the same group, that is $group[x, g]$ and $group[father[x], g]$ are both equal to one, which requires several constraints to be modeled using ILP (see [19] for a modeling of the product of two binary variables). Constraint (7) states that the path from a direct child of $sink$ to $sink$ cannot be broken, as $sink$ is always active (*i.e.*, in all groups). Finally, Constraint (8) states that for the other nodes x, the path from x to $sink$ is broken if the link $(x, father[x])$ is broken, or if the path from $father[x]$ to $sink$ is broken.

The model could be improved by considering that each link $(x, father[x])$ has a quality $q_x \in]0; 1]$. This quality has an impact on the number of retransmissions required by x so that $father[x]$ receives the packet. Reducing the congestion is thus related to reducing the number of retransmissions, rather than distributing the nodes in each group. The impact on this model would be the following: (i) the meaning of $brokenLink[x]$ and $brokenPath[x]$ would change to represent the number of expected transmissions (and retransmissions) to forward a packet from x to the sink, (ii) the variable $brokenLink[x]$ would be weighted by a function of q_x, q_x being a constant (fixed for each x) in the ILP, and (iii) the computation of $brokenPath[x]$ for $x \neq sink$ and $father[x] = sink$ would also be weighted by a function of q_x. Note that the other computations of variable $brokenPath[x]$ would not change, as the number of retransmissions from x to the sink would still be equal to the number of retransmissions on the link $(x, father[x])$ plus the number of retransmissions from $father(x)$ to the sink.

2) Heuristic solution: We now propose our heuristic, described in Algorithm 1, that allows us to compute the groups efficiently (although not optimally). Our heuristic starts by distributing nodes to the $m \geq 3$ groups using the same greedy approach as the one used in Subsection III-A (except that it is applied to $m \geq 3$ groups instead of 2). Then, for each group that has less than the intended minimum number of nodes, the heuristic requests nodes to the largest subtree t_{max} such that (i) the root of t_{max} is a child of the sink and (ii) the root of t_{max} is in the largest group g_{max}.

The heuristic to request $requested$ nodes from the subtree of a group g_{big} rooted at a node r, is described in Algorithm 2. If node r is a leaf and is still in g_{big}, it can switch to the requesting group g_{small}, resulting into one node obtained from g_{big}. If node r is a leaf but is not in g_{big} anymore, r has already been switched to another group, and cannot switch again. If node r is a branching node, all children of r are considered in a random order (and the node r itself is considered at last). For each child $child$ of r, the number of requested nodes is proportional to the size of the subtree of $child$, and also depends on the remaining number of nodes that have to be obtained.

C. Protocol description

In this subsection, we derive a protocol from our heuristic based on local information. The number of groups m and the minimum size of each group min are the two parameters of the protocol. The protocol uses two rounds. In the first round, each node computes its number of descendants on the routing tree. Once this number is obtained, the protocol divides nodes into m

minimize $\sum_{x \in N} brokenPath[x]$

such that $\forall x \in N$, $\sum_{g \in [1;m]} group[x,g] = 1$	(1)
$\forall g \in [1;m]$, $\sum_{x \in N} group[x,g] \geq min$	(2)
$\forall x \in N$, $\forall g \in [1;m]$, $sameGroup[x,g] \leq group[x,g]$	(3)
$\forall x \in N$, $\forall g \in [1;m]$, $sameGroup[x,g] \leq group[father[x],g]$	(4)
$\forall x \in N$, $\forall g \in [1;m]$, $sameGroup[x,g] \geq group[x,g] + group[father[x],g] - 1$	(5)
$\forall x \in N$, $x \neq sink$, $brokenLink[x] = 1 - \sum_{g \in [1;m]} sameGroup[x,g]$	(6)
$\forall x \in N$, $x = sink$ or $father[x] = sink$, $brokenPath[x] = 0$	(7)
$\forall x \in N$, $x \neq sink$ and $father[x] \neq sink$, $brokenPath[x] = brokenLink[x] + brokenPath[father[x]]$	(8)

Table I
INTEGER LINEAR CONSTRAINTS FOR A SCHEDULE OF $m \geq 3$ GROUPS.

Algorithm 1 Construction of a schedule of $m \geq 3$ groups.

Require: t is the routing tree rooted at the sink, min is the minimum size of a group
  $l \leftarrow$ empty list
  for each child $child$ of the sink **do**
    add subtree rooted at $child$ to list l
  end for each
  sort list l in decreasing order of size of the subtrees
  for each subtree t_{child} of l **do**
    $g \leftarrow$group having the minimum number of nodes
    add all nodes of subtree t_{child} into g
  end for each
  $g_{max} \leftarrow$ group having the largest number of nodes
  $t_{max} \leftarrow$ largest subtree rooted at a child of the sink, such that the root of t_{max} is in group g_{max}
  for each group g **do**
    if g has less than min nodes **then**
      request enough nodes from t_{max} to reach min nodes (see Algorithm 2)
    end if
  end for each

Algorithm 2 Transfer of nodes from g_{big} to g_{small}.

Require: $requested$ is the number of requested nodes, r is the root of the considered subtree, g_{small} is the group requesting nodes, g_{big} is the requested group
Returns the number of nodes obtained from g_{big}
  if r is a leaf **then**
    if r is in g_{big} **then**
      r switches to g_{small}
      return 1
    else
      return 0
    end if
  else
    $sum \leftarrow 0$
    for each each child $child$ of r (in random order) **do**
      $x \leftarrow requested * size(child)/size(r)$
      request x nodes from subtree rooted at $child$ (recursively)
      $requested \leftarrow requested-$number of nodes obtained
      $sum \leftarrow sum+$number of nodes obtained
    end for each
    if $requested > 0$ and r is in g_{big} **then**
      r switches to g_{small}
      return $sum + 1$
    else
      return sum
    end if
  end if

groups according to the greedy heuristic (described in Subsection III-A for $m = 2$, and in the first part of Algorithm 1 in Subsection III-B for $m \geq 3$). In the second round (required only when $m \geq 3$), the sink sends a request for nodes from groups having less than min nodes to nodes of another group.

The first round is performed by having the sink send a `count-descendants` message to each of its children. When a branching node receives this message, it forwards it to all of its children. When a leaf node receives this message, it sends a `count-descendants-reply` message with a value of one to its father. When a branching node has received the `count-descendants-reply` messages from all of its children, it sums up all the values, and sends the total value (plus one for itself) to its father. At the end of the first round, each node knows the number of descendants for each of its children.

The second round is performed by following Algorithm 2. Notice that in this algorithm, each node r uses only local information, except for $size(child)$ (which is the number of descendants of $child$) that has been computed in the first round. Each recursive call to the algorithm is implemented by sending a `request-nodes` message to the given child. Each return from a call is implemented by sending a `reply-with-nodes` message to the father of the node, with the number of nodes that have switched to the new group.

It can be noticed that our protocol requires few messages per node, and only modifies the schedule of node activations, rather than the channel access mechanism or the routing protocol.

The assumption that routes are static (and, hence, that the tree t is static too) can be weaken: if $m \geq 3$, the protocol is still able to operate if routes change, although the overall network performance might decrease. When a route changes, it is possible that a node u becomes in a

different group from its father v: in this case, the route from u to the sink is broken, as well as the routes from all the descendants of u to the sink. However, u can still send data to its new father v, although it might have to wait depending on the activation periods. Consequently, we assume that the sink maintains the total number of nodes affected by route changes that have occurred. When this number exceeds a threshold, the protocol recomputes new groups based on the new routing tree (which requires the sink to restart the two rounds), and resets the counter of nodes affected by route changes.

IV. Simulation results

In this section, we present our simulation settings and we highlight the results obtained using our heuristic and protocol described in Section III. We compare our results to those obtained with ZigBee.

A. Simulation settings

Our simulation environment is NS-2. In our simulations, we considered for simplicity reasons a set of 100 nodes distributed on a 10×10 grid, as shown on Figure 4. Nodes are distant of 10 m from their neighbors on the grid. The sink is node 99, which is at the top-right corner of the grid. The propagation model we use is the shadowing model with the following parameters: the path loss exponent is set to 3, and the random variable is a Gaussian variable with a mean of zero and a standard deviation of 3. Nodes transmit with a power of -5 dBm, and have a reception threshold of -85 dBm, which is the minimum receiver sensitivity of the standard IEEE 802.15.4. These settings follow the measurements reported in [20]. Our simulator implements interferences, collisions and capture effects. Each node has a queue of 50 packets.

The basic approach is ZigBee (with the default routing protocol, which is AODV), and uses for the MAC layer the IEEE 802.15.4 standard in beacon-enabled mode. It is considered as our basic approach as it does not provide any scheduling, and we used the same medium access mechanism. The beacon interval (value BI of the standard) is set to 6, which yields a cycle duration of about one second (983 ms). For all approaches, we varied the duty cycle α within $\{0.15, 0.25, 0.35\}$. The superframe duration SD is computed as $\alpha.BI$. There is no collision free period in our scenario. The packet size is 30 bytes (at the MAC layer, that is without the PHY overhead). Each node (except the sink) is a source: all nodes produce one data packet at every period (that is why we considered packets with small payload). We varied the period of packet generation from 1 s to 10 s, with each node starting its packet generation at a random time within the first period. Finally, each simulation lasts for 100 seconds, and results are averaged over 100 repetitions.

Figure 4 shows the routing links between nodes as parent-child relationships. Those links were generated by AODV [11], and were fixed for the whole duration of the simulation (so that the routing is static). Another routing protocol, such as the hierarchical routing protocol of ZigBee [10] or OLSR [21] could be used instead of the tree depicted on Figure 4. However, we decided to keep this tree as it shows a situation which is a worst-case for our heuristic and protocol. Indeed, notice that child 88 of the sink has a subtree containing many nodes (there are 88 nodes in the subtree). Using a tree that is more balanced would simplify the construction of the schedule, and improve our results.

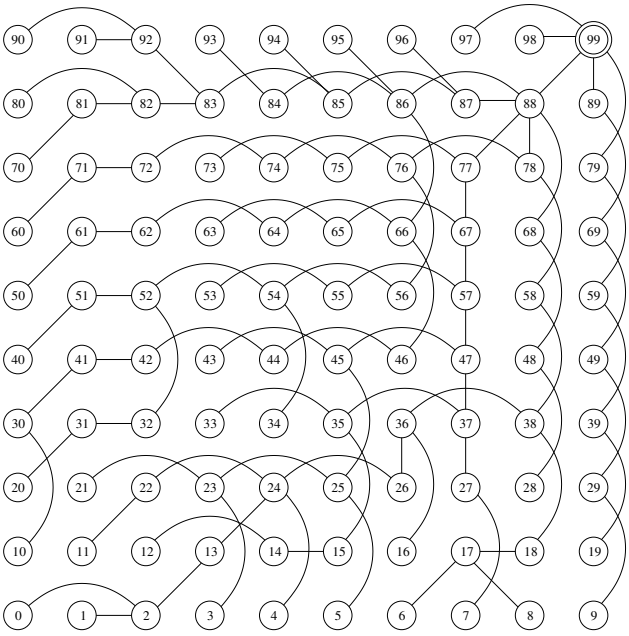


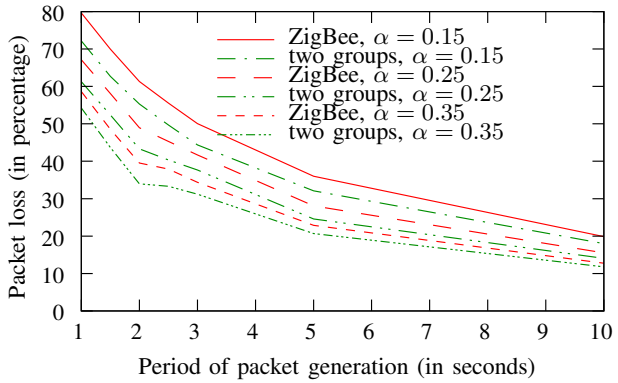
Figure 4. Example of a routing tree (obtained using AODV) over a grid topology: lines represent parent-child relationships. This tree is used in all our simulations.

In the following, we compare four protocols: ZigBee, our heuristic with $m = 2$ groups, the solution obtained using our ILP, and our heuristic with $m = 3$ groups. For the ILP, the minimum group size is set to 10, and the groups obtained from the ILP are integrated into NS-2. For the heuristic with $m = 3$, the minimum group size is also set to 10.

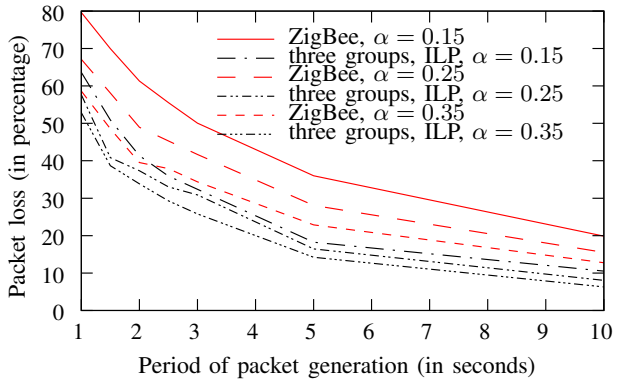
B. Packet loss

Figure 5 shows the performance of all the protocols in terms of packet loss. We refer to the packet loss as $(n_g - n_r)/n_g$, where n_g is the total number of packets generated by the sources and n_r is the total number of packets received by the sink. Thus, the packet loss metric takes into account both the losses due to collisions and to channel access failures.

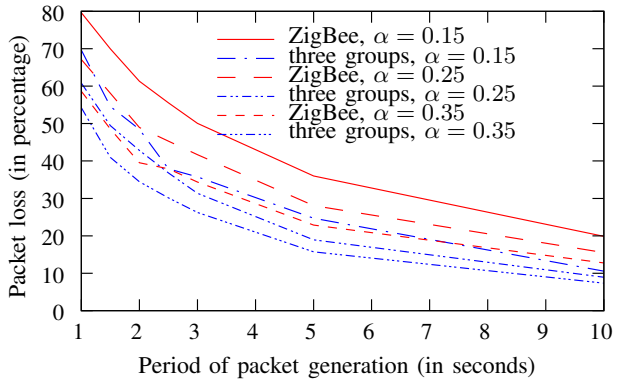
1) Heuristic with two groups: Figure 5(a) shows the percentage of packet loss as a function of the period of the packets generated per source. For instance, a value of 5 on the x-axis means that every source generates one packet every 5 seconds. For this period, the total number of generated packets is $99 \times 100/5 = 1980$ packets (as there are 99 sources and our simulation lasts for 100



(a) Comparison between ZigBee and the heuristic for two groups.



(b) Comparison between ZigBee and the ILP.



(c) Comparison between ZigBee and the heuristic for three groups.

Figure 5. Performance of the protocols in terms of packet loss as a function of the packet generation.

seconds). We can notice that the packet loss is large when the period of generation is small, and decreases with the number of generated packets, as expected. Indeed, when the network is overloaded, collisions appear frequently in the network, which yields to several congested areas. Moreover, the packet loss decreases when the duty cycle increases. This is due to the fact that when the duty cycle increases, nodes are active for more time, and thus, they can route more packets. Our heuristic shows better performance compared to ZigBee: the gain is about 10% for $\alpha = 0.15$, about 9% for $\alpha = 0.25$, and about 8% for $\alpha = 0.35$ (for a period of 1). This gain is due to the fact that our heuristic splits nodes into two groups and thus during a given period, only nodes of one of these two groups are able to send and receive packets, which decreases congestion in the network (including around the sink). Notice that on the AODV tree we used for simulations, all the nodes of the largest subtree are in the first group and the remaining nodes are in the second. As a result, group 1 contains 88 nodes and group 2 contains 11 nodes. The gain of our heuristic would be larger with a tree having more balanced subtrees.

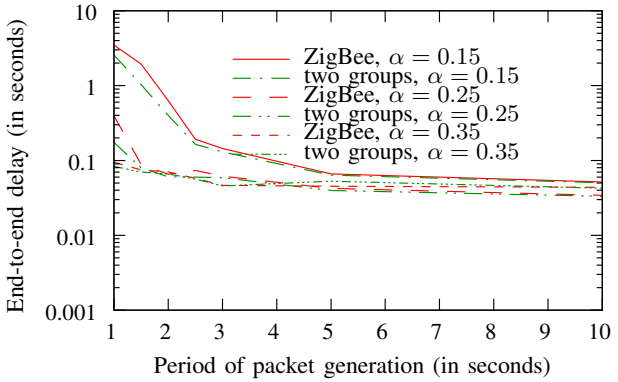
2) ILP with three groups: Figure 5(b) shows the percentage of packet loss as a function of the period of packets generated per source. The gain the ILP achieves compared to ZigBee, for a period of 1, is 20% for $\alpha = 0.15$, 15% for $\alpha = 0.25$, and 10% for $\alpha = 0.35$. For a period of 10, the gain varies between 47% and 51%. Note that 8% of the paths were broken by the ILP.

3) Heuristic with three groups: Figure 5(c) shows the percentage of packet loss in terms of the period of generated packets per source. The heuristic breaks 9% of paths in average. The gain that our heuristic achieves compared to ZigBee, for a period of 1, is 13% for $\alpha = 0.15$, 11% for $\alpha = 0.25$, and 8% for $\alpha = 0.35$. For a period of 10, the gain varies between 42% and 50% for all values of α. Thus, the maximum gain of the ILP solution compared to our heuristic is about 37% for $\alpha = 0.15$, 30% for $\alpha = 0.25$, and 20% for $\alpha = 0.35$. The gain compared to ZigBee is due to the fact that our heuristic produces a schedule which is almost similar to the ILP schedule in this scenario, and thus the network performance (packet loss and end-to-end delay) is improved.

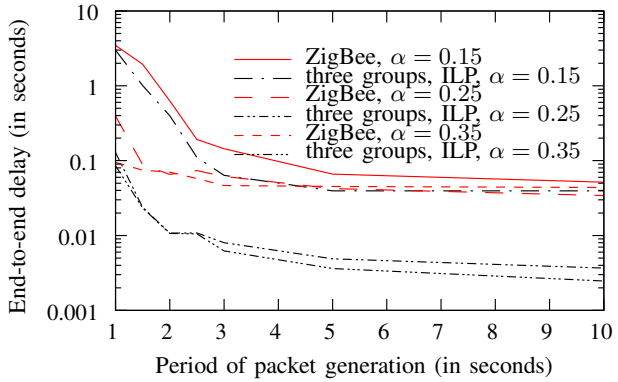
C. Delay

Figure 6 shows the performance of all the protocols in terms of delay. We refer to the end-to-end delay as the time duration experienced by a packet from its generation at the source to its reception at the sink. The end-to-end delay only takes into account the packets that are correctly received by the sink. We chose to present the delay using a logarithmic scale on the y-axis in order to show the behavior of the two approaches, even for large periods.

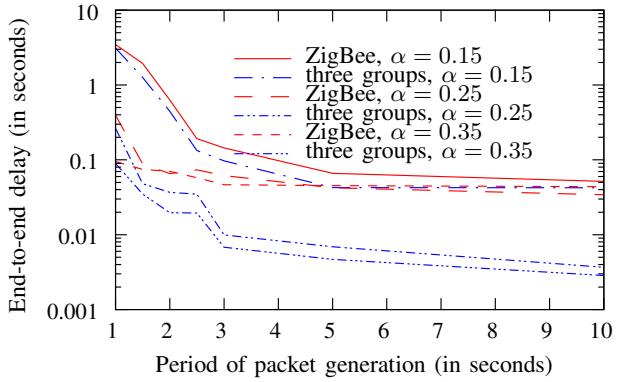
1) Heuristic with two groups: Figure 6(a) illustrates the average end-to-end delay as a function of the period of the packets generated per source. We can notice that the delay decreases with the period of generated packets: the less packets, the less delay, as congestion increases the CSMA/CA backoffs and the number of channel access attempts, as well as the number of the packet in queue. Furthermore, the figure shows the impact of the duty cycle α on the delay. For a small value of α, the delay is more important than for a large value. When comparing our heuristic with ZigBee for a period of 1, the gain is 27% for $\alpha = 0.15$, 56% for $\alpha = 0.25$, and 12% for $\alpha = 0.35$. For a period of 10, the gain varies between 2% and 3% for all values of α. This variation in gain comes from several phenomenons. For large α, the congestion of the network is low, and thus the impact of our approach is reduced. For small values of α, our approach benefits from less congestion, but is penalized by having the delay computed for more packets than ZigBee.



(a) Comparison between ZigBee and the heuristic for two groups.



(b) Comparison between ZigBee and the ILP.



(c) Comparison between ZigBee and the heuristic for three groups.

Figure 6. Performance of the protocols in terms of delay as a function of the packet generation.

2) ILP with three groups: Figure 6(b) shows the average end-to-end delay as a function of the period of generated packets per source. For a period of 1 packet generated per second and per source, the gain the ILP achieves compared to ZigBee is 14% for $\alpha = 0.15$, 69% for $\alpha = 0.25$, and 10% for $\alpha = 0.35$. It can be noticed that the results obtained by our heuristic for two groups are better than the results obtained by the ILP for three groups. This is due to the fact that no path is broken when two groups are considered (if paths were broken with two groups, some nodes would not be able to communicate with their next hop), but 8% of the paths are broken for $m = 3$. For a period of 10 (low traffic load), the gain reaches 41% for $\alpha = 0.15$, 91% for $\alpha = 0.25$, and 94% for $\alpha = 0.35$. This shows that the approach has some potential, as an optimal schedule can achieve good results.

These results, as well as those from Figure 5(b), confirm that it is generally better to activate nodes according to a schedule, even if the packets on few paths might suffer from a large delay when a node has to wait for the activation of its next hop. Indeed, in our scenario, the gain obtained by reducing the congestion outperforms the loss in delay, as the number of broken paths is small.

3) Heuristic with three groups: Figure 6(c) shows the average end-to-end delay as a function of the period of generated packets per source. The behavior of the end-to-end delay is almost the same as the one obtained by the ILP schedule. The gain our heuristic achieves compared to ZigBee varies between 11% and 18% for $\alpha = 0.15$ (and all the periods), between 35% and 89% for $\alpha = 0.25$, and between 5% and 90% for $\alpha = 0.35$. The gain of the ILP solution compared to our heuristic reaches 25% for $\alpha = 0.15$, 3% for $\alpha = 0.25$, and 1% for $\alpha = 0.35$. These results show again that it is better to split nodes into groups provided that few paths are broken. This approach reduces congestion in network which improves the network performance in terms of packet loss and end-to-end delay without increasing the energy consumption as nodes are always active for the same duration.

D. Other metrics

The energy required by our approach is similar as the energy required by ZigBee. Indeed, as explained in Section III, nodes are active during a percentage α of the time in both cases. The only difference in energy consumption comes from the number of activations/deactivations of the radio module: our approach requires $2m$ activations/deactivations per cycle (with m the number of groups), while ZigBee only requires 2 per cycle. However, we consider that the energy consumption required to activate/deactivate the radio module is negligible, as this process is very fast in usual IEEE 802.15.4-compliant components (as the CC2420 for instance [22]) compared to the duration of the activity of the node per cycle.

The cost of our protocol in terms of number of broken paths and control messages exchanged is presented on Table II. The first column is the minimum number of nodes intended per group. Note that in some conditions, our protocol cannot find enough nodes to request from the largest group, and thus it is possible that for some routing trees, some groups have less than the intended minimum size after the end of the second round of the protocol. This behavior was not observed in our simulations, however.

We considered $m = 3$ groups and we varied the minimum group size. We notice that for groups of 5 nodes minimum, there is no difference between both solutions. The number of broken paths increases with the minimum number of nodes that a group can contain for both solutions: ILP and heuristic. This is due to the fact that, by increasing the required number of nodes per group, some nodes have to be requested from a larger group and several nodes become in a different group than their next hop. We notice also that the performance of our

heuristic stays close to the ILP performance in terms of broken paths.

The last column shows the number of messages exchanged by each node during the computation of groups. The number of control messages is low and increases slowly with the minimum number of nodes per group. The larger the groups, the more control messages are exchanged to balance the groups. We also notice that for all minimum group sizes, the number of control messages per node is limited and thus the energy consumption of our protocol is low.

Table II
OVERHEAD OBTAINED BY THE ILP AND BY OUR HEURISTIC.

Minimum group size	Number of broken paths ($m = 3$)		Number of control messages per node
	ILP	heuristic	
5	0	0	1.98
10	8	9	2.64
15	18	20	3.28
20	28	36	3.94

V. CONCLUSIONS

Generally, in synchronized protocols for WSNs such as ZigBee, all nodes are active simultaneously. In this paper, we study how to distribute nodes into groups, such that nodes are activated depending on their group. We propose an exact modeling of the problem based on integer linear programming, as well as an heuristic that yields to results close to the optimal. Then, we propose a protocol that can implement this heuristic with a limited number of control messages. We compare our propositions to ZigBee and we show that our protocol can significantly reduce congestion. Simulation results show that our solution outperforms ZigBee in terms of packet loss rate and end-to-end delay. For a period of 1 and all values of α, for instance, the gain in packet loss varies between 8% and 12%, and the gain in end-to-end delay varies between 5% and 35%, without changing the energy consumption.

REFERENCES

[1] K. Al Agha, M.-H. Bertin, T. Dang, A. Guitton, P. Minet, T. Val, and J. Viollet, "Which wireless technology for industrial wireless sensor networks? the developpement of OCARI technology," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 10, pp. 4266–4278, 2009.

[2] G. Boggia, P. Camarada, L. A. Griecp, and M. R. Palattella, "Fire detection using wireless sensor networks: An approach based on statistical data modeling," in *NTMS (New Technologies, Mobility and Security)*. IEEE, 2008, pp. 1–5.

[3] G. Werner-Allen, K. Lorincz, M. Ruiz, and O. Marcillo, "Deploying a wireless sensor network on an active volcano," in *Internet Computing*, vol. 10, no. 2. IEEE, 2006, pp. 18–25.

[4] IEEE 802.15, "Part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (WPANs)," ANSI/IEEE, Standard 802.15.4 R2006, 2006.

[5] B. Yahya and J. Ben-Ohtman, "Towards a classification of energy aware MAC protocols of wireless sensor networks," *WCMC (Wireless Communications and Mobile Computing)*, vol. 9, no. 12, pp. 1572–1607, 2009.

[6] E. Dashkova and A. Gurtov, "Survey on congestion control mechanisms for wireless sensor networks," in *Internet of Things, Smart Spaces, and Next Generation Networking*, ser. LNCS 7469, 2012, pp. 75–85.

[7] J. Kim, J. On, S. Kim, and S. Lee, "Performance evaluation of synchronous and asynchronous MAC protocols for wireless sensor networks," in *SensorComm (Sensor Technologies and Applications)*. IEEE, 2008, pp. 500–506.

[8] A. K. Azad, H. Kabir, and M. B. Hossain, "A survey on schedule-based MAC protocols for wireless sensor networks," *International Journal of Computer Science and Network*, vol. 2, pp. 120–128, 2013.

[9] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *SenSys (Embedded networked sensor systems)*. ACM, November 2004, pp. 95–107.

[10] ZigBee, "ZigBee Specification," ZigBee Standards Organization, Standard Zigbee 053474r17, January 2008.

[11] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (AODV) routing," IETF, Request For Comments 3561, July 2003.

[12] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Infocom (Annual Joint Conference of the IEEE Computer and Communications Societies)*, vol. 3. IEEE, 2002, pp. 1567–1576.

[13] P. Huang and L. Xiao, "TAS-MAC: A traffic-adaptive synchronous MAC protocol for wireless sensor networks," in *SECON (Sensor, Mesh and Ad Hoc Communications and Networks)*. IEEE, 2013, pp. 113–121.

[14] M. Abdul Hamid, M. Abdullah-Al-Wadud, and I. Chong, "A schedule-based multi-channel MAC protocol for wireless sensor networks," *Sensors*, vol. 10, no. 10, pp. 9466–9480, 2010.

[15] G. P. Halkes and K. G. Langendoen, "Crankshaft: An energy-efficient MAC-protocol for dense wireless sensor networks," in *EWSN (European Wireless Sensor Networks)*, ser. LNCS 4373, 2007, pp. 228–244.

[16] P. Suriyachai, J. Brown, and U. Roedig, "Time-critical data delivery in wireless sensor networks," in *DCOSS (International Conference on Distributed Computing in Sensor Systems)*, ser. LNCS 6131, 2010, pp. 216–229.

[17] J. Degesys, I. Rose, A. Patel, and R. Nagpal, "DESYNC: Self-organizing desynchronization and TDMA on wireless sensor networks," in *IPSN (Information Processing in Sensor Networks)*. IEEE, 2007, pp. 11–20.

[18] R. M. Karp, "Reducibility among combinatorial problems," *Complexity of Computer Computations*, pp. 85–103, 1972.

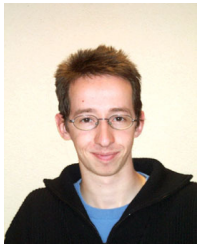
[19] A. Billionnet, *Optimisation discrète*, ser. InfoPro. Dunod, 2007, EAN13: 9782100496877.

[20] C. Gezer, C. Buratti, and R. Verdone, "Capture effect in IEEE 802.15.4 networks: modelling and experimentation," in *ISWPC (International Symposium on Wireless Pervasive Computing)*. IEEE, 2010, pp. 204–209.

[21] C. Adjih, T. Clausen, P. Jacquet, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L. Viennot, "Optimized link state routing protocol," IETF, RFC 3626, 2003.

[22] "CC2420 - 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF transceiver," Chipcon, Preliminary datasheet revision 1.2, 2004.

Nancy El Rachkidy is an assistant professor at Clermont Université, Université Blaise Pascal, France. She is doing her research at LIMOS-CNRS. She received her PhD in 2011 at Université Blaise Pascal. She obtained her MSc degree in networks and computer science from the Lebanese University of Beirut, Lebanon, in 2006. Her research interests include wireless communincations, sensor network, MAC and routing protocols.



Alexandre Guitton is an assistant professor at Clermont Université, Université Blaise Pascal, France. He is doing his research at LIMOS-CNRS. He received his PhD in 2005 and his MSc in 2002 at University of Rennes I, in the field of computer networks. He has been working at Clermont Université since 2007. His research interests include wireless communications, sensor networks, MAC protocols, and energy-efficiency.