Clustering and Communications Scheduling in WSNs using Mixed Integer Linear Programming

François Avril, Thibault Bernard, Alain Bui, and Devan Sohier

Abstract: **We consider the problem of scheduling communications in wireless sensor networks (WSNs) to ensure battery preservation through the use of the sleeping mode of sensors. We propose a communication protocol for 1-hop WSNs and extend it to multi-hop WSNs through the use of a 1-hop clustering algorithm. We propose to schedule communications in each cluster in a virtual communication ring so as to avoid collisions. Since clusters are cliques, only one sensor can speak or listen in a cluster at a time, and all sensors need to speak in each of their clusters at least once to realize the communication protocol. We model this situation as a mathematical program.**

Index Terms: **1-hop clustering, collision avoidance in WSNs, communication scheduling.**

I. INTRODUCTION

Sensors are tiny battery-powered devices that scan a number of environmental parameters and wirelessly communicate between them to monitor phenomena pertaining to a wide range of applications such as: Environmental issues, fire disasters, military applications, etc. Data collected by sensors are relayed to a base station (BS) in charge of analyzing it and triggering an alarm if needed.

The most critical requirements of wireless sensor networks (WSNs) are concerned with the following three aspects: Routing (i.e., how to transmit the scanned data efficiently to the base station), security (i.e., how to ensure that scanned data remain anonymous and cannot be used by a third party), and power-saving (i.e., how to reduce battery consumption and increase the lifetime of each sensor).

In this study, we focus on the third aspect, that is, saving power and increasing the network lifetime while ensuring communications in the WSN. Several studies have focused on techniques for relaying data efficiently in order to reduce energy consumption [1], [2] or to guarantee that messages are delivered to the intended recipients [3], [4]. We focus on collision avoidance in this work. Indeed, as shown in [5], most of the energy is consumed in a WSN for communications between sensors. A collision causes the sensor to retransmit the message, wasting energy. In order to minimize collisions, the wake-up/sleep mechanisms and/or the control messages RTS/CTS/ACK, defined in 802.11x standards, are used to design energy-efficient MAC protocols for WSNs such as those described in T-MAC [6], Z-MAC [7], X-MAC, and B-MAC[8].

Manuscript received April 11, 2014

François Avril, Alain Bui, and Devan Sohier are with the CaRO Team PRiSM CNRS, Université de Versailles Saint-Quentin-en-Yvelines, 45, av. des Etats Unis F-78035 Versailles Cedex, France, email: {francois.avril, alain.bui, devan.sohier}@prism.uvsq.fr.

Thibault Bernard is with the CaRO Team PRiSM CNRS, Université de Versailles Saint-Quentin-en-Yvelines, 45, av. des Etats Unis F-78035 Versailles Cedex , France and with SYSCOM Team CReSTIC, Université de Reims Champagne Ardenne F-51687 Reims, Cedex, France, email: thibault.bernard@univ-reims.fr.

Digital object identifier 10.1109/JCN.2014.000072

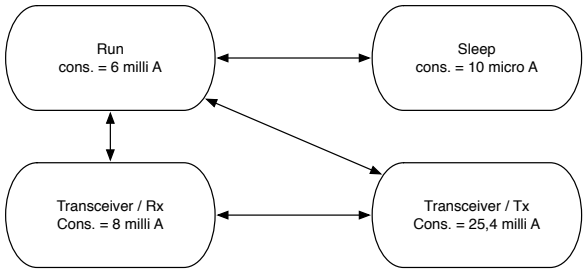


Fig. 1. Energy consumption of Mica 2 sensor at different states.

In this paper, we propose to save battery power by scheduling communications so that sensors spend as much time as possible in their *sleep* state. In [9] and [10], we have proposed to structure communications among all sensors in a 1-hop network in such a way that no collision is possible. We propose to make sensors of a cluster communicate in a (virtual) token ring so that only one sensor sends data at a time, and only one other sensor receives them (i.e., all other sensors are in the *sleep* state). In [11], we have proposed a preliminary clustering procedure to extend this protocol to multi-hop WSNs.

In this paper, we propose a complete clustering procedure that divides the WSN in 1-hop clusters and investigate on scheduling the discussion among all clusters to avoid collisions, in particular on *gateway* sensors (sensors belonging to several clusters).

This paper is organized as follows, Section II presents related works about energy preservation in WSNs. In Section III, we present the model used to build the scheduling solution and previous works on which our solution rely. In Section IV, we define an adequate clustering and provide an algorithm to split a multi-hop WSN into 1-hop clusters. An example is provided in Section V to illustrate correct scheduling.

In Section VI, a mixed integer linear programming (MILP) description of the scheduling problem is presented and some optimization issues are analyzed. Concluding remarks and future works are given in Section VII.

II. RELATED WORKS

Battery preservation in WSNs is a major concern, and many solutions have been proposed to address it In [5], the authors describe typical energy consumption (cf. Fig. 1) of sensors depending on their state. They have shown that power is mainly spent for communications between sensors. Thus maximizing

the time spent in the *sleep* mode preserves battery and increases sensor lifetime. In addition, it avoids collisions that are expensive in terms of energy consumption. The protocol we propose aims at increasing the time spent by sensors in the sleeping mode and avoiding collisions.

In [12], the authors proposed to maximize the lifetime of a non-mobile WSN through routing procedures. They formulated the routing problem as a linear programming problem, where the objective is to maximize the network lifetime. Their algorithm is based on the *link cost* of communication between sensors: They build shortest paths for routing. They save energy by considering the information flow exchanged between sensors. In our solution, we focus not on the size of the transmitted information but on the discussion scheduling over the sensors.

In [13], the authors proposed minimizing power use by reducing the transmission power of each sensor. As the consumed power increases quadratically with the transmission range, they developed an approximation algorithm of a connected minimal weighted dominated set, and used it as a backbone over sensors (by reducing the transmission radius), while ensuring the connectivity of the whole network. This algorithm could be applied in a preliminary phase to our solution.

WSNs use synchronization for TDMA scheduling, data fusion, sleep period synchronization, etc. The problem of synchronization in WSNs has been investigated in [14]. In the authors words, "*existing time synchronisation methods were not designed with wireless sensor networks in mind, and need to be extended or redesigned.*" The authors designed an algorithm in the context of WSNs (time computation and bandwidth constraints) to achieve synchronization. Their solution has not been specifically applied for discussion scheduling.

In [15], the authors proposes a technique called "linear distance-based scheduling" of the *sleep/running* mode of every sensor to reduce power use. This technique is designed to work in a clustered network: The further from the cluster-head a sensor is, the greater the probability that this sensor is in the *sleep* mode. Thus, the power use of every sensor depends on the distance to the clusterhead. This creates a discrepancy between sensors with respect to the battery preservation, and leads to connectivity loss. As a conclusion, the authors proposed to use a dynamic cluster formation to balance power consumption discrepancies.

In [16], the authors addressed the problem of wake-up schedule in WSNs where 1-hop and 2-hop neighbors may interfere with sensor's communications. To avoid this, a sensor should avoid using the same slot as those used in its 2-hop neighborhood. The authors proposed a tree-based schedule in a distributed way for data collection.

In [17], the authors presented a survey on optimization techniques to address several problems in WSNs, such as coverage, topology control, mobility, scheduling and routing. In particular, authors noted that in several investigations, the time-slot allocation problem (scheduling) is formulated as a graph-coloring problem, to model the fact that two edges adjacent to the same sensor cannot use the same time slot.

We propose in this paper a scheduling mechanism of the discussion between clusters. We express the different constraints between clusters as a mixed integer linear system. An introduction to mathematical programming can be found in [18]. A general framework to deal with a wide class of periodic scheduling problems is presented in [19].

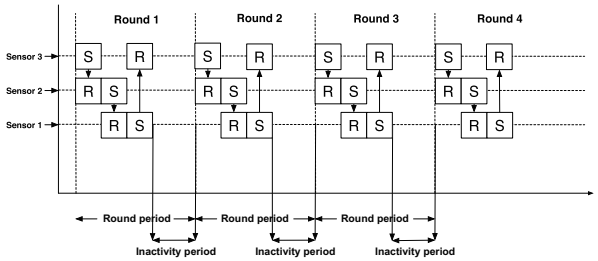


Fig. 2. Discussion protocol among sensors.

III. MODELS AND PRELIMINARY WORK

A. Network Model

We aim at managing communications over a WSN made up of n sensors identified as $s_1, \cdots, s_n$.

We make the following assumptions:

- Each sensor has a unique identifier.
- The communication graph of the sensors is connected and undirected.
- Sending a message is instantaneous.
- All sensors are operating at the same speed.

The system consists of n independent sensors, that communicate over a wireless medium. Such a WSN can be represented as an undirected graph $G = (V, E)$, where V is a set of sensors and E is a set of edges that represent possible communications between sensors, i.e., (i, j) belongs to E means that i can communicate with j (and conversely).

In a given graph $G = (V, E)$, we denote by $n = |V|$ the number of sensors in the network. Each sensor executes the same algorithm. The neighborhood N_i of a sensor i is defined as $N_i = \{j \in V | (i, j) \in E\}$. The number of neighbors of a sensor i is called its degree, denoted by $\deg(i)$.

B. Preliminary Works

In this section, we present the 1-hop discussion protocol described in [9] and [10].

B.1 1-hop Discussion Protocol

We proposed in [9] a light 1-hop discussion protocol over a clique of sensors. In this protocol, communication relies on a ring communication structure. Each sensor has two slots for communication: one for receiving data (Slot R), and another for sending data (Slot S). Sensors wake up periodically two by two so that, at a given time, there is at most one sensor sending data and another receiving data. Fig. 2 describes the basic procedure of the protocol. During a period each sensor wakes up in such way as to receive information from its predecessor, and then deliver information to its successor. The remaining time is spent in the *sleep* mode.

The protocol is set up in a distributed manner: Sensors agree to establish the discussion in an ordered way. Sensors first synchronize and then compute a schedule in the ring, that sums

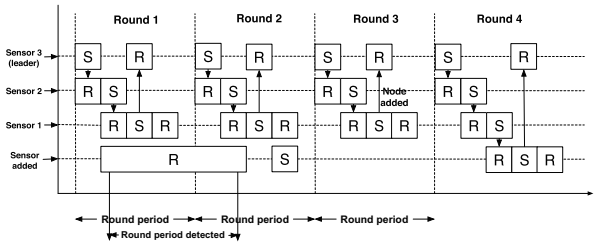


Fig. 3. Adding a sensor in the discussion.

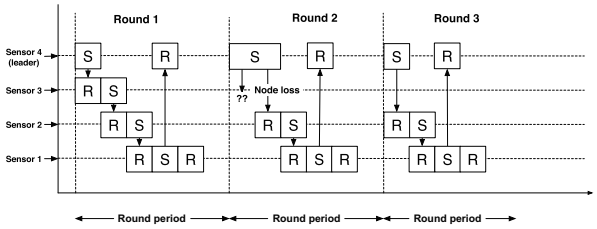


Fig. 4. Sensor crash.

up to an ordering. This protocol is designed for applications in which no alert needs to be reported urgently. This protocol avoids periodical negotiation phases like those used in beacon-enabled IEEE 802.15.4 protocols such as the ZigBee protocol.

B.2 Refinement with the Joining/Leaving Procedure of Sensors

In [10], we further refined the protocol to allow the insertion or the deletion of sensors in the round discussion.

To include new sensors in the discussion, we use the remaining time period at the end of the round (the time period during which every sensor is sleeping). Fig. 3 summarizes the joining procedure for a sensor.

During the first round, a joining sensor has to keep listening in order to detect several parameters of the discussion between sensors such as the round period length and the last receiving slot. The sensor announces at the end of the second round that it will join the network. As shown in Fig. 3, the last sensor's slot is longer than that of the other sensors. This additional amount of time is allocated for listening to sensors willing to join the network. Thus, in the second round, the last sensor is informed about a new sensor connection, and it waits for the third round to inform the first sensor about this connection. During the third round, all sensors are informed of the connection and, at the end of this round, the joining sensor is considered as a network member. Its slot will be larger than that of the others in order to be able to manage the connection of an extra sensor. This method balances power use. Indeed, the last sensor of the ring consumes more energy than others, and the last sensor is the one that has joined the network the most recently (thus, it should have more energy than other sensors).

The updated protocol also handles a sensor's disconnection. Fig. 4 summarizes the entire procedure when a sensor disconnects from the network.

When a sensor crashes, the discussion has to be updated to ensure continuous information propagation. The disconnection will be detected with a basic acknowledgement mechanism:

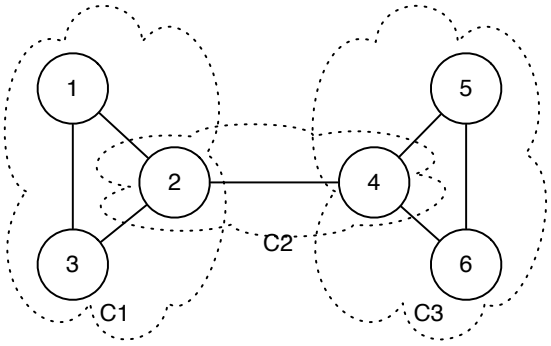


Fig. 5. A clique-clustered sensors network.

When an acknowledgement is missing, the sending sensor keeps awake until another sensor is able to receive the frame (obviously, the next uncrashed sensor in the ring). The slots assigned to all other sensors in the remaining discussion are then shifted. The parameters of the shift procedure depend on the number of crashed sensors and their locations in the round discussion.

IV. CLUSTERING ALGORITHM FOR MULTI-HOP WSNS

In this section, we propose a clustering algorithm allowing this token ring procedure to be implemented on each cluster. The schedule of the communication between clusters will be discussed in Section VI. An earlier version of this clustering technique can be found in [11].

A. Specification of the Clustering

The token ring protocol presented in [9] requires a complete communication graph (i.e., a 1-hop WSN). Therefore the clustering we propose computes cliques.

A sensor can be involved in several clusters: For instance, in Fig 5, three clusters have been computed and Sensor 2 and Sensor 4 belong respectively to clusters $(C1, C2)$ and $(C2, C3)$. These sensors are called *gateway* sensors.

Such sensors are involved in several rounds of 1-hop communications, more precisely one round for each cluster they belong to. To schedule the discussion rounds between clusters, two *adjacent* clusters (i.e., clusters that share a set of common sensors) cannot be involved in communications at the same time, as it would imply message collisions. To take these constraints into account, the computed clustering is such that two adjacent sensors always belong to a common cluster, more precisely the edge between these two sensors has to be included in at least one cluster.

To limit collisions between clusters, and consequently the constraints in scheduling, we build maximal cliques. A maximal clique is a clique that is included in no larger clique. The algorithm presented below computes all maximal cliques in the 2-hop neighborhood of a node, and then discards the cliques that are not necessary to cover all edges.

In order to deliver all the environmental data scanned by the sensors to the base station, the connectivity of the graph has to

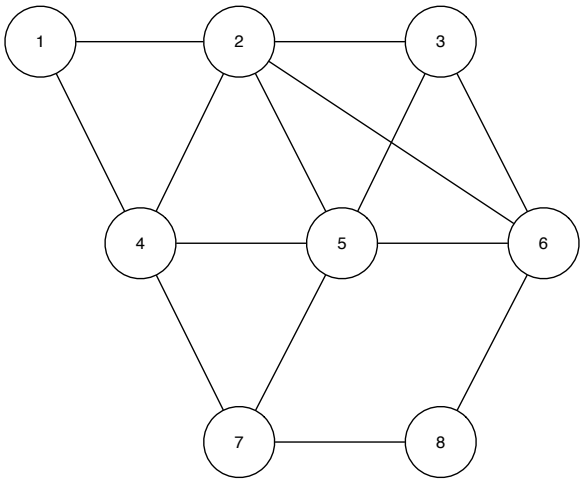


Fig. 6. A wireless sensor network and its clustering.

be maintained throughout the clustering procedure.

This gives rise to the following requirements:

- **Covering:** Every edge has to be covered by at least one cluster (so that constraints on communications of the two sensors are taken into account).
- **Connectivity:** The graph composed by all sub-graphs of all clusters is connected.
- **Maximal clique:** All sensors in the same cluster are adjacent, and no other sensor is adjacent to all sensors in the cluster.

With these requirements and a correct schedule of the wake-up/sleeping state between clusters, the 1-hop discussion protocol can be extended to multi-hop.

B. Clique-Clustering Algorithm

Each sensor computes cliques by exchanging messages, and locally decides the set of clusters to which it belongs. The result of the clustering procedure is a set of cluster $\mathcal{C}$.

The algorithm has three phases:

- **The setup phase** is the phase during which a sensor successively broadcasts its id and its neighborhood. When receiving this data from its neighbors, the sensor updates local variables. At the end of this phase, each sensor knows its 2-hop neighborhood.
- **The clique computation phase** is the phase when each sensor computes from its local variables all cliques to which it belongs.
- **The cleaning phase** is the final phase and consists in discarding clusters c such that all edges contained in c also belong to another cluster of $\mathcal{C}$.

The three phases of the clustering procedure are explained on the example given in Fig. 6.

B.1 2-hop Neighborhood Computation

At the end of phase 1 (Algo. 1), we have the following results on the network given Fig. 6. We deliberately omit variables $\mathrm{Neigh}_{i,j}$ since $\mathrm{Neigh}_{i,j}$ is Neigh_j as stored on sensor i. Starting from then, all sensors know the ids of their neighbors, and we do not distinguish between a sensor and its id any longer.

Algorithm 1 Setup phase on sensor i

```
Neigh_i ← ∅
for all j ∈ N_i do
    Neigh_{i,j} ← ∅
end for
2hop_Neigh_i ← ∅
for all j ∈ N_i do
    {Inform all neighbors of one's id}
    Send id_i to j
end for
waiting ← N_i
while waiting ≠ ∅ do
    {Receive information on neighbors' ids}
    Receive id_j from a neighbor j
    Neigh_i ← Neigh_i ∪ {id_j}
    waiting ← waiting \ {j}
end while
for all j ∈ N_i do
    {Inform all neighbors of one's neighborhood}
    Send Neigh_i to j
end for
waiting ← N_i
while waiting ≠ ∅ do
    {Receive information on neighbors' neighborhoods}
    Receive Neigh_j from a neighbor j
    Neigh_{i,j} ← Neigh_j
    for all k ∈ Neigh_j do
        if k ∉ 2hop_Neigh_i then
            2hop_Neigh_i ← 2hop_Neigh_i ∪ {k}
        end if
    end for
    waiting ← waiting \ {j}
end while
```

Var.\ Sens.	1	2	3
Neigh	2,4	1,3,4,5,6	2,5,6
2h-Neigh	1,2,3,4,5,7	1,2,3,4,5,6,7,8	1,2,3,4,5,6,7,8

Var.\ Sens.	4	5	6
Neigh	1,2,5,7	2,3,4,6,7	2,3,5,8
2h-Neigh	1,2,3,4,5,6,7,8	1,2,3,4,5,6,7,8	1,2,3,4,5,6,7

Var.\ Sens.	7	8
Neigh	4,5,8	6,7
2h-Neigh	1,2,3,4,5,6,7	2,3,4,5,6,8

To compute its 2-hop neighborhood, a sensor sends two messages (one with its id, and another with its 1-hop neighborhood) to its neighbors, and receives $2\deg(i)$ messages from its neighbors.

B.2 Cluster Computation

Algorithm 2 computes maximal cliques by taking an edge (j,k) in the 2-hop neighborhood and adding sensors l in the 2-hop neighborhood of i if they are adjacent to all sensors of the computed clique.

This algorithm computes all clusters to which the sensor belongs or to which it is adjacent (information required for the cleaning phase). As the clusters must cover edges, the sensor

Algorithm 2 Clique computation phase on sensor i

1: **for all** $j \in 2\text{hop\_Neigh}_i$ **do**
2: **for all** $k \in \text{Neigh}_{i,j}$ **do**
3: $C_{j,k} \leftarrow \{j,k\}$
4: {$C_{j,k}$ is a clique containing edge (j,k)}
5: **for all** $l \in \text{Neigh}_{i,j}$ **do**
6: {A sensor l adjacent to all sensors in $C_{j,k}$ is added to the clique}
7: $Ok \leftarrow true$
8: **for all** $m \in C_{j,k}$ **do**
9: **if** $m \notin \text{Neigh}_{i,l}$ **then**
10: $Ok \leftarrow false$
11: **end if**
12: **end for**
13: **if** Ok **then**
14: $C_{j,k} \leftarrow C_{j,k} \cup \{l\}$
15: **end if**
16: **end for**
17: {$C_{j,k}$ is then maximal}
18: $\mathcal{C} \leftarrow \mathcal{C} \cup C_{j,k}$
19: **end for**
20: **end for**
21: {$\mathcal{C}$ contains maximal cliques containing any edge in the 2-hop neighborhood of sensor i}

computes a maximal clique C_{jk} for each edge (j,k) in its 2-hop neighborhood. The algorithm adds to the cluster each sensor l adjacent to all sensors in the cluster.

At the end of the second phase of the clustering procedure in the example shown in Fig. 6, $\mathcal{C} = \{\{1,2,4\}, \{2,4,5\}, \{2,3,5,6\}, \{4,5,7\}, \{6,8\}, \{7,8\}\}$.

Consider δ an upper bound on the degree of sensors. As a clique cannot contain more than δ sensors (all sensors in a clique being neighbors), the outer loop has a range in $O(\delta^2)$ and each of the three other loops has a range in $O(\delta)$. Hence, the complexity of this algorithm is $O(\delta^5)$.

B.3 Cleaning Phase

The cleaning phase aims at discarding clusters that cover no edge that is not covered by other clusters. Indeed, such clusters do not provide information on potential collisions that is not already known. As clusters are maximal cliques, no cluster can be included in another one; but a cluster can be included in the union of several other clusters, hence, such a cluster is discarded.

For instance, in the previous example, clique $\{2,4,5\}$ can be discarded because edges $(2,4)$, $(2,5)$, and $(4,5)$ belong respectively to cliques $\{1,2,4\}$, $\{2,3,5,6\}$, and $\{4,5,7\}$.

For every clique C containing i, the algorithm checks if every pair of sensors (l,m) (also an edge) is also part of another clique D. If this is the case, then the clique is discarded.

At the end of the clustering procedure, for the example given in Fig. 6, $\mathcal{C} = \{\{1,2,4\}, \{2,3,5,6\}, \{4,5,7\}, \{6,8\}, \{7,8\}\}$.

The complexity of this algorithm is upper-bounded by $O(\delta^5)$ ($O(\delta)$ for loop at line 1, and $O(\delta^2)$ for loops at lines 3 and 5).

Algorithm 3 Cleaning phase on sensor i

1: **for all** $C \in \mathcal{C}$ with $i \in \mathcal{C}$ **do**
2: $discard \leftarrow true$
3: **for all** $(l,m) \in C^2$ with $l \neq m$ **do**
4: $found \leftarrow false$
5: **for all** $D \in \mathcal{C}, D \neq C \wedge \neg found$ **do**
6: **if** $(l,m) \in D$ **then**
7: $found \leftarrow true$
8: **end if**
9: **end for**
10: **if** $\neg found$ **then**
11: {C cannot be discarded because edge (l,m) is not covered by any other clique}
12: $discard \leftarrow false$
13: **end if**
14: **end for**
15: **if** $discard$ **then**
16: {C can be discarded because all its edges are covered by other cliques}
17: $\mathcal{C} \leftarrow \mathcal{C} \backslash C$
18: **end if**
19: **end for**

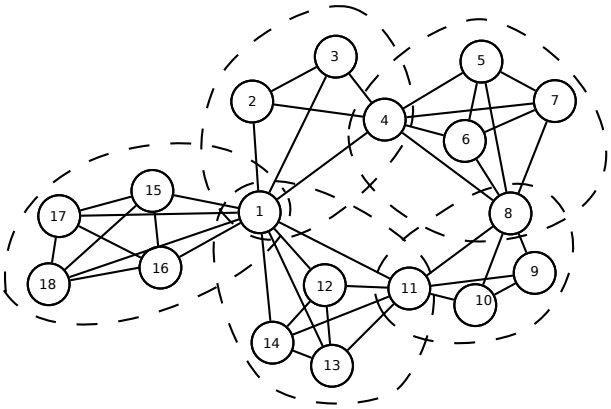


Fig. 7. Communication graph of a WSN.

V. SCHEDULING: AN EXAMPLE

The example given below illustrates the result of the clustering, and the constraints it imposes on the scheduling of communications.

We consider the network presented in Fig. 7.

First, we apply the clustering algorithm presented in Section IV on this network. Computed clusters are $A = \{1,2,3,4\}$, $B = \{4,5,6,7,8\}$, $C = \{8,9,10,11\}$, $D = \{1,11,12,13,14\}$, and $E = \{1,15,16,17,18\}$.

We can notice that, in a cluster, the role of non-gateway sensors is symmetric, i.e., the place of two non-gateway sensors of a cluster, such as Sensor 2 and Sensor 3, can be exchanged.

A valid communication planning on this network is shown in Fig. 8.

In this example, cluster A and cluster C can communicate at the same time since they have no common sensor. Cluster B and D are in a similar situation.

Here, we choose to place communications of cluster A and

$n\backslash t$	1	2	3	4	5	6	7	8	9	10	11
1	R	S			R	S				R	S
2		R	S								
3	S			R							
4			R	S				R	S		
5					S				R		
6							R	S			
7						R	S				
8	R	S			R	S					
9		R	S								
10	S			R							
11			R	S				R	S		
12						R	S				
13							R	S			
14					S				R		
15									R	S	
16								R	S		
17							R	S			
18							S				R

Fig. 8. Scheduling on the network.

$n\backslash t$	1	2	3	4
1	R	S		
2		R	S	
3	S			R
4			R	S

Fig. 9. Scheduling of cluster A.

$n\backslash t$	1	2	3	4
8	R	S		
9		R	S	
10	S			S
11			R	S

Fig. 10. Scheduling of cluster C.

C first, followed by communications of clusters B and D, and finally, communications of cluster E.

For cluster A, the first two time slots are reserved for communications of sensor 1, which is a gateway sensor with cluster D. Similarly, we start communications of cluster C with communications of gateway sensor 8. As sensors 4 and 8 are in cluster B, they cannot be involved in communications at the same time. Thus, sensor 4 can speak in cluster A only at time slots 3 and 4. For the same reason, sensor 11 can only communicate at time slots 3 and 4. This leads to the scheduling presented in Figs. 9 and 10.

We build the scheduling for cluster B and D, presented in Figs. 11 and 12 in the same fashion. As sensor 4 is involved in communication at time 4, sensors of cluster B cannot communicate before time slot 5; similarly, sensors of cluster D cannot communicate because of sensor 11.

Finally, we need to place communications of cluster E. As the sensors of cluster D communicate until time slot 9, sensor 1 cannot communicate until time slot 10. Therefore, we build a

$n\backslash t$	5	6	7	8	9
4				R	S
5	S				R
6			R	S	
7		R	S		
8	R	S			

Fig. 11. Scheduling of cluster B.

$n\backslash t$	5	6	7	8	9
1	R	S			
11				R	S
12		R	S		
13			R	S	
14	S				R

Fig. 12. Scheduling of cluster D.

$n\backslash t$	7	8	9	10	11
1				R	S
15			R	S	
16		R	S		
17	R	S			
18	S				R

Fig. 13. Scheduling of cluster E.

planning algorithm for cluster E with communication of sensor 1 taking place in time slots 10 and 11 (Fig. 13).

The combination of all these plannings gives the solution shown in Fig. 8.

VI. CLUSTER COMMUNICATION SCHEDULING

Distant clusters can communicate at the same time. However, the presence of gateway sensors poses a problem in planning communications, and forbids straightforward application of the token algorithm. Indeed, if two clusters share a gateway, its communication according to the 1-hop protocol for any of these clusters prevents communications among sensors of the others: If a sensor has to listen to another given sensor, no other sensor within hearing distance must speak.

Making all sensors communicate one after the other is a valid communication scheme, but this scheduling is long and not scalable.

In this section, we model constraints on communications defining the validity of a schedule. First, we present decision variables, then we present constraints that define the 1-hop protocol on a cluster. After that, we present constraints that come from the existence of gateway sensors, and discuss the objective function of such a planning.

A. Variables

The scheduling is based on:

- $G = (V, E)$: The communication graph of the system and
- $\mathcal{C}$: The set of clusters computed by the clustering algorithm (a cluster being a set of sensors).

We consider T an upper bound on the optimal duration of a planning. As we can build a valid communication scheduling with every sensor in every cluster speaking one after the other, we choose $T = \sum_{c\in\mathcal{C}} |c|$, where $|c|$ is the number of sensors in c (considering that the time unit is one slot, $|c|$ is the time to achieve one round discussion in the cluster c).

We also define the following decision variables, the instantiation of which defines a scheduling:

- $\forall c \in \mathcal{C}, \forall t \in \{0,\cdots,T\}, \forall i \in C,$

$$e_i^c(t) = \begin{cases} 1, \text{ if } i \text{ is in sending mode at time } t \text{ for cluster } c; \\ 0, \text{ else.} \end{cases}$$

- $\forall c \in \mathcal{C}, \forall t \in \{0,\cdots,T\}, \forall i \in C,$

$$r_i^c(t) = \begin{cases} 1, \text{ if } i \text{ is in reception mode at time } t \text{ for cluster } c; \\ 0, \text{ else.} \end{cases}$$

- $\forall c \in \mathcal{C}, \forall i \in C,$

$$s_i^c = \begin{cases} 1, \text{ if } i \text{ starts communication for cluster } c; \\ 0, \text{ else.} \end{cases}$$

We use the convention that $\forall c \in \mathcal{C}, \forall i \in c, \forall t \leqslant 0, e_i^c(t) = 0$.

B. Intra-Cluster Constraints

As the first sensor speaking in a cluster in a round is the cluster initiator, and as, according to the protocol described in section III-B, a sensor speaks just after listening (except for the initiator, which speaks first), variables $r_i^c(t)$ can be deduced from $e_i^c(t)$ and $s_i^c(t)$. A non-initiator sensor i that is in reception mode would in sending mode in the following time slot. In this case, $s_i^c = 0$ and $r_i^c(t) = e_i^c(t+1)$. If i is the initiator, it is in sending mode $|c|$ time slot before being in reception mode. In this case, $s_i^c = 1$, and $e_i^c(t-|c|) = r_i^c(t)$.

Thus, $r_i^c(t) = (1 - s_i^c) \times e_i^c(t+1) + s_i^c \times e_i^c(t-|c|)$.

Now, this equation is not linear, and to be able to use a MILP solver, we replace it with an equivalent set of linear equations or inequations.

First, as all quantities are nonnegative, we have:

$$\begin{cases} r_i^c(t) \geqslant (1 - s_i^c) \times e_i^c(t+1), \\ r_i^c(t) \geqslant s_i^c \times e_i^c(t-|c|). \end{cases}$$

As either $s_i^c(t)$ or $1 - s_i^c(t)$ is zero, these inequations imply that $r_i^c(t) \geq (1 - s_i^c) \times e_i^c(t+1) + s_i^c \times e_i^c(t-|c|)$, and as all sensors speak and listen exactly once in each of their cluster (constraints stated later will ensure this), the above-mentioned constraints are equivalent to $r_i^c(t) = (1 - s_i^c) \times e_i^c(t+1) + s_i^c \times e_i^c(t-|c|)$.

Now, for a and b in $\{0,1\}$, we have $a.b \geq a + b - 1$. Thus

$$\begin{cases} r_i^c(t) \geqslant (1 - s_i^c) + e_i^c(t+1) - 1 = e_i^c(t+1) - s_i^c, \\ r_i^c(t) \geqslant s_i^c + e_i^c(t-|c|) - 1. \end{cases}$$

Conversely, the last equations are equivalent to the previous ones since $r_i^c(t) \geq 0$.

This leads to the two following sets of linear constraints:

$$\forall c \in \mathcal{C}, \forall i \in c, \forall t \in \{0,\cdots,T\},\ r_i^c(t) \geqslant e_i^c(t+1) - s_i^c. \quad (1)$$

$$\forall c \in \mathcal{C}, \forall i \in c, \forall t \in \{0,\cdots,T\},\ r_i^c(t) \geqslant s_i^c + e_i^c(t-|c|). \quad (2)$$

There is at most one sensor in sending mode at a time in a cluster:

$$\forall c \in \mathcal{C}, \forall t \in \{0,\cdots,T\}, \sum_{i\in c} e_i^c(t) \leqslant 1. \quad (3)$$

And there is at most one sensor in reception mode at a time in a cluster:

$$\forall c \in \mathcal{C}, \forall t \in \{0,\cdots,T\}, \sum_{i\in c} r_i^c(t) \leqslant 1. \quad (4)$$

For a cluster c, all sensors in c are in sender mode exactly once (for c) during a round, implying that for any sensor i, there is an unique time slot t such that i is in sending mode:

$$\forall c \in \mathcal{C}, \forall i \in c,\ \sum_{t=0}^{T} e_i^c(t) = 1. \quad (5)$$

All sensors are in receiver mode exactly once:

$$\forall c \in \mathcal{C},\ \forall i \in c,\ \sum_{t=0}^{T} r_i^c(t) = 1. \quad (6)$$

When a sensor is in sending mode for a cluster, there is at least one sensor in reception mode for the cluster (if sensor i is not sending data, i.e., $e_i^c(t) = 0$, the constraint below is not restrictive, and constraint 4 already imposes that at most one sensor is listening in the cluster):

$$\forall c \in \mathcal{C},\ \forall i \in c, \forall t \in \{0,\cdots,T\}, e_i^c(t) \leqslant \sum_{j\in c} r_j^c(t). \quad (7)$$

Similarly, when a sensor is in reception mode, there is at least one sensor in sending mode for the cluster:

$$\forall c \in \mathcal{C},\ \forall i \in c, \forall t \in \{0,\cdots,T\}, r_i^c(t) \leqslant \sum_{j\in c} e_j^c(t). \quad (8)$$

C. Inter-Cluster Constraints

Inter-cluster constraints model the fact that, when a gateway sensor is speaking in one of its clusters, no adjacent sensor (i.e., no sensor in another of its clusters) can be listening; and that, on the other hand, when it is listening in a cluster, no adjacent sensor can be speaking.

When a sensor i belongs to several clusters, if i is in sending mode for one cluster, no sensor in the other clusters to which i belongs can be in reception mode:

$$\forall c \in \mathcal{C}, \forall i \in c, \forall t \in \{0,\cdots,T\},$$
$$\sum_{\substack{c' \neq c \\ c' \ni i}} \sum_{j\in c'} r_j^{c'}(t) \leqslant (1 - e_i^c(t)) \times n \times |\mathcal{C}|. \quad (9)$$

$n \times |\mathcal{C}|$ is obviously always greater than $\sum_{c'\neq c, c'\ni i} \sum_{j\in c'} r_j^{c'}(t)$, since $r_j^{c'}(t)$ is always less than 1. Thus, this constraint only imposes that when sensor i is speaking in c (i.e., $e_i^c(t) = 1$), $\sum_{c'\neq c, c'\ni i} \sum_{j\in c'} r_j^{c'}(t) = 0$, i.e., that all $r_j^{c'}(t) = 0$ (since

$r_j^{c'}(t) \geq 0$); thus no sensor j in a cluster c' other than c and to which i belongs can be listening.

Similarly, if i is in reception mode for one cluster, no sensor in other clusters to which i belongs can be in sending mode:

$$\forall c \in \mathcal{C}, \forall i \in c, \forall t \in \{0, \cdots, T\},$$
$$\sum_{\substack{c' \neq c \\ c' \ni i}} \sum_{j \in c'} e_j^{c'}(t) \leqslant (1 - r_i^c(t)) \times n \times |\mathcal{C}|. \tag{10}$$

Constraints 1 through 10 define a communication scheduling such that all sensors listen and speak exactly once in each of their clusters, and no two communications can collide.

D. Objective Function and Interpretation

Any scheduling verifying constraints 1 through 10 is valid. However, we are interested in computing an optimal scheduling according to some criterion. This criterion is expressed through an objective function, which we try to minimize (or to maximize).

We minimize the round length. This objective may, for instance, describe a network in which local information updates need to be as frequent as possible.

To describe this situation, we introduce a new variable Y that is an upper bound on the duration of the scheduling. For any time t, if there is a $e_i^c(t) \neq 0$, $Y \geqslant t$. Hence, if $e_i^c(t) = 1$, we have $Y \geqslant t \times e_i^c(t)$. Else, if $e_i^c(t) = 0$, this constraint is $Y \geq 0$, which is always true.

This leads to the following constraints:
$\forall t \in \{0, \cdots, T\}, \forall c \in \mathcal{C}, \ \forall i \in c,$

$$Y \geqslant t \times e_i^c(t). \tag{11}$$

Next, we seek to minimize Y, the smallest upper bound, which is the length of the shortest planning. This leads to the objective function:

$$\min\{Y\}. \tag{12}$$

Variables $e_i^c(t)$, $r_i^c(t)$, s_i^c, and Y, constraints 1 through 11, and the objective function $\min(Y)$ make up a mathematical program that can be solved by integer linear programming. Solutions of such a program are computed by a solver like CPLEX and give the shortest possible scheduling in a given network.

Solving a MILP is a NP-hard problem. Hence, this method requires (at worst) a number of elementary operations that is exponential in the number of variables and constraints. This centralized method (that is not designed to be implemented on sensors) computes optimal planning and is designed to provide lower bounds on schedules that sensors may be able to compute. In future, we intend to work on more economical decentralized methods to compute sub-optimal schedules.

VII. CONCLUSION

In this paper, we investigate the computation of a communication schedule that allows a maximum of parallel communications in a clustered multi-hop WSN. We develop a mixed integer linear program representing a centralized method to determine the optimal scheduling. The specific clustering procedure that allows this scheduling is also presented.

Such an optimal scheduling, though computed in a centralized fashion, is useful to compute benchmarks to which distributed solutions can be compared.

As future works, we intend to work on distributed methods to compute schedules that avoid message collisions, wherein the entire network will organize itself by local arrangement. Through local organization, sensors have to design an efficient mechanism for gathering information until it reaches the base station. In particular, an arborescent hierarchization of the schedule will be investigated.

REFERENCES

[1] J. Ben-Othman and B. Yahya, "Energy efficient and qos based routing protocol for wireless sensor networks," *J. Parallel Distrib. Comput.*, vol. 70, no. 8, pp. 849–857, 2010.

[2] B. Yahya and J. Ben-Othman, "Towards a classification of energy-aware MAC protocols for wireless sensor networks," *IWCMC*, vol. 12, no. 3, pp. 1572–1607, 2009.

[3] Y. M. Akkaya Kemal, "A survey on routing protocols for wireless sensor networks," *Ad Hoc Netw.*, vol. 3, no. 5, pp. 325–349, 2005.

[4] A. Djouama, L. Mokdad, and M. Abdennebi, "Performance evaluation of lifetime-driven admission control for infrastructure-less clustered wireless networks," *J. CCPE*, vol. 25, no. 5, pp. 718–727, 2013

[5] A. da Cunha and D. da Silva Jr., "An approach for the reduction of power consumption in sensor nodes of wireless sensor networks: Case analysis of mica2," in *Proc. Workshop on Embedded Computer Systems: Architectures, Modeling, and Simulation, LNCS 4017*, 2006, pp. 132–141.

[6] Y. Wei, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *Proc. INFOCOM*, 2002, pp. 1567–1576.

[7] I. Rhee *et al.*, "Z-mac: An hybrid mac for wireless sensor networks," in *Proc. SenSys*, 2005, pp. 90–101.

[8] M. Buettner *et al.*, "X-mac: A short preamble mac protocol for duty-cycled wireless sensor networks," in *Proc. SenSys*, 2006, pp. 307–320.

[9] T. Bernard and H. Fouchal, "Slot assignment over wireless sensor networks," in *Proc. IEEE GLOBECOM*, 2011, pp. 1–5.

[10] T. Bernard and H. Fouchal, "A low energy consumption mac protocol for WSN," in *Proc. IEEE ICC*, 2012, pp. 533–537.

[11] K. Abdurusul,T. Bernard, and H. Fouchal, "An efficient multi-hop MAC protocol for WSN," in *Proc. International Symposium on Computers and Communications*, 2012.

[12] J. H. Chang and L. Tassiulas, "Maximum lifetime routing in wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol.12, no. 4, pp. 609–619, 2004.

[13] J. Ben-Othman *et al.*, "Self-stabilizing algorithm for efficient topology control in wireless sensor networks," *J. Computational Sciences*, vol. 4, no. 4, pp. 199–208, 2013.

[14] M. L. Sichitiu and C. Veerarittiphan, "Simple, accurate time synchronization for wireless sensor networks," in *Proc. WCNC*, 2003, pp. 1266–1273.

[15] J. Deng *et al.*, "Scheduling sleeping nodes in high density cluster-based sensor networks," *J. Mobile Netw. Appl.*, vol. 10, no. 6, pp. 825–835, 2005.

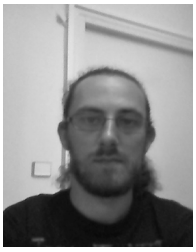
[16] F. J. Wu and Y. C. Tseng, "Distributed wake-up scheduling for data collection in tree-based wireless sensor networks," *IEEE Commun. Lett.*, vol. 13, no. 11, pp. 850–852, 2009.

[17] A. Gogu *et al.*, "Optimization problems in wireless sensor networks," in *Proc. CISIS*, pp. 302–309, 2011.

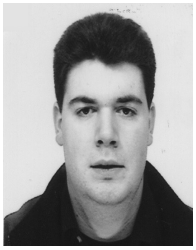
[18] L. Wolsey, *Integer Programming.* New York: Wiley, 1998.

[19] P. Serafini and W. Ukovich, "A mathematical model for periodic scheduling problems," *J. Discret. Math.*, vol. 2, no. 4, pp. 550–581, 1989.

[20] A. Gogu, D. Nace, and Y. Challal, "A framework for solving the configuration problem in wireless sensor networks," in *Proc. BWCCA*, 2013, pp. 64–60.



Francois Avril obtained in 2010 his M.S. degree in Mathematics from the University of Science of Reims in France. He is currently a Teaching Assistant and Ph.D. student at University of Versailles Saint-Quentin-en-Yvelines under the supervision of Dr. Devan Sohier and Prof. Alain Bui. His research is centered on stochastics algorithms for distributed system.



Thibault Bernard is currently Associate Professor at the Université de Reims Champagne Ardenne and a Member of the PRiSM Computer Science Lab. He received the Ph.D. degree in Computer Science in 2006. His research interests are distributed algorithms, random walks, fault tolerance, and ad hoc networks.

Alain Bui is Full Professor of Computer Science PRiSM-CNRS Laboratory at the University of Versailles Saint-Quentin, France. Alain Bui received his Ph.D. degree in Computer Science in 1994 from université Paris 7 and INRIA. His interests include distributed algorithms, random walks, self-stabilization, and operations research.

Devan Sohier Devan Sohier is currently Associate Professor at the Computer Science Department of Versailles University and a Member of the PRiSM Computer Science Lab. He defended his Ph.D. Thesis at the Ecole Pratique des Hautes Etudes in 2005, following which he was Associate Professor for 3 years at Reims University. His research topics include operations research (optimization under uncertainties) and distributed computing (particularly the design and study of random walk based distributed algorithms in systems subject to topological changes). He advises three Ph.D. students, two of whom defended in June 2011 and Dec. 2012.