
GETRF delivrable 2:

# Joint Routing and TDMA-based Scheduling to Minimize Delays in Grid Wireless Sensor Networks

Ichrak Amdouni, Cedric Adjih, Pascale Minet
Hipercom2 Team
Inria Paris-Rocquencourt

February 2014

# Contents

**1 Overview about the Document and the Contributions** **3**

**2 Context** **5**

**3 Background** **7**
3.1 Delay Efficient TDMA . . . . . 7
3.2 VCM: Vector-based Coloring Method . . . . . 7

**4 Assumptions and Problem Statement** **9**
4.1 Assumptions . . . . . 9
4.2 Problem Statement . . . . . 9

**5 Principles and Overview about the Solution** **10**
5.1 Discussion about Intuitive Solutions . . . . . 10
5.1.1 First solution: OPERA [7] . . . . . 10
5.1.2 Second solution: Spanning Tree with VCM . . . . . 10
5.2 Principles . . . . . 11
5.3 Overview . . . . . 12

**6 *Routes*: Dominating Tree Construction** **15**
6.1 Principles of the Construction of a Dominating Tree . . . . . 16
6.2 Algorithm of the Construction of a Dominating Tree . . . . . 17
6.3 Properties of a Dominating Tree . . . . . 17
6.3.1 General Properties . . . . . 17
6.3.2 Properties Concerning Nodes in $\mathcal{P}_A$ . . . . . 20
6.4 Results: Examples of Dominating Trees . . . . . 21
6.5 Routing Over the Dominating Tree . . . . . 22

**7 CO-Routes: Colors Ordering for *Routes*** **22**
7.1 Method of CO-Routes . . . . . 22
7.2 Results about the Delays Obtained . . . . . 23

**8 *Highways* Construction and Coloring** **24**
8.1 *Highways* Construction and Routing over *Highways* . . . . . 24
8.2 CO-Highways: Colors Ordering for *Highways* . . . . . 25

**9 Orchestration in a TDMA cycle** **26**

**10 Conclusion** **27**

# 1 Overview about the Document and the Contributions

This document describes our contribution in the GETRF project as far as the network coloring applied to sensor networks is concerned. In this project, we focus on TDMA based wireless networks and introduce sensor node activity scheduling solutions based on graph coloring. So far, we have targeted the optimization of the energy consumption. In this document, in addition to the gain in energy saving, we focus on another metric: the end-to-end delays. Optimizing delays is very important especially in emergency applications as in almost all surveillance applications targeted in the GETRF project. Hence, our objective is to minimize the data collection delays. Our strategy consists in integrating coloring with routing: First, we start from a colored grid with a periodic coloring (coloring that is obtained by the repetition of a color pattern). Then, routes are built taking into account these colors. Finally, colors of nodes are ordered in the TDMA cycle following the order of appearance of these nodes in routes. Our contributions are summarized as follows:

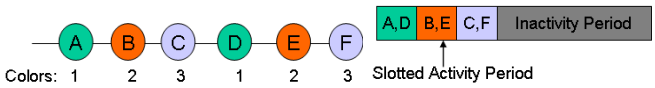
- In general, the delay optimization problem is either managed by routing or by scheduling separately. In our work, we address this problem by a cross-layer approach illustrated by the integration of the scheduling and the routing.
- Usually researchers consider that minimizing delays in TDMA based medium access schemes means minimizing the TDMA cycle length. In addition to this reason, we believe that given a route, the scheduling order of the nodes on this route is also crucial (as a packet delivery might be delayed when a node receives a packet after its slot). Hence, in our work, we consider both problems.
- Given the grid network, we introduce a hierarchical routing architecture where sensor nodes route data to a predefined set of data aggregators and these aggregators route data to the data sink.
- The produced delays are optimized: the integration of the coloring with the routing allows any node to reach the closest aggregator in a single TDMA cycle.
- Also, any node is able to reach a subset of aggregators other than the closest one in a single cycle. This ensures fault tolerance in case of failure of the closest aggregator. Also, it is useful in a scenario of multiple sinks or mobile sink.

- Paths between aggregators are also fast: less than or equal to one cycle.

The remaining of this document details the background and the solutions proposed as well as the results obtained.

# 2 Context

TDMA scheduling is one of the energy efficient techniques used in WSNs. Each node is allocated a time slot for data transmissions. The main drawback of this pure TDMA is the underutilization of the bandwidth. To limit this problem, many solutions providing bandwidth spatial are proposed such as graph coloring based solutions. Generally those solutions work as follows. The graph is colored such that no two nodes that have the same color interfere. As an example, let us consider the linear network depicted in Figure 1(a). Assuming that interferences are limited to 2 hops, the same color is assigned to nodes $A$ and $D$, while a second color is assigned to nodes $B$ and $E$ and a third color is assigned to nodes $C$ and $F$ because these couples of nodes are 3 hops away. Consequently, they can transmit in the same slot without interfering (see cycle in Figure 1(b)). The slots form the activity




(a) Sample linear network. (b) TDMA cycle.

Figure 1: Example of a network and the TDMA cycle associated.

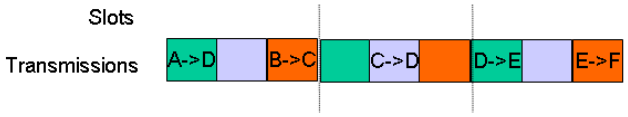
period in the TDMA cycle (also called superframe). The remaining part of the TDMA cycle corresponds to the inactivity period where all nodes can turn off their radio and save energy. Consequently, this scheme avoids energy wasted in idle listening and collisions.

While the main advantage of TDMA in WSNs is energy saving, it also allows the determination of bounds on end-to-end delays. Indeed, contention-based protocols suffer from collisions especially under high data rate. This factor makes the end-to-end delays unpredictable. For delay sensitive applications like fire detection, this is not acceptable. However, it does not suffice to assign slots to nodes in an arbitrary way to obtain minimum end-to-end delays. Two reasons may increase delays:

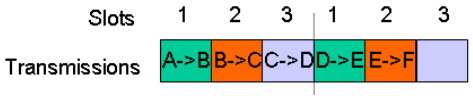
- **First reason**: Due to the fact that interfering nodes cannot have the same slot, the cycle obtained may be very long. A schedule of minimum length can be achieved by maximizing the reuse of time slots. Therefore, most existing algorithms aim at maximizing the number of transmissions scheduled in the same slot and enable spatial reuse by devising strategies to eliminate interferences [1, 2].

- **Second reason**: While routing protocols address the delay issues by choosing the shortest path, the order of medium access by nodes on

this path is also crucial. Let us consider the example of Figure 1(a) and the two schedules depicted in Figures 2(a) and 2(b). In the second schedule, colors purple and orange are associated with slots 2 and 3 respectively instead of slots 3 and 2 in the first schedule. Notice that, consequently, slots assigned to $E$ and $F$ are switched.




(a) First ordering: green→slot1, purple→slot2, orange→slot3.




(b) Second ordering: green→slot1, orange→slot2, purple→slot3.

Figure 2: Examples of two color ordering for the same network.

Following the first scheduling (Figure 2(a)), node $A$ requires 9 time slots to transmit data to node $F$. However, if nodes follow the second schedule (Figure 2(b)), only 5 slots are required. The reason is that, in the first scheduling, node $B$ receives the packet from $C$ after the end of its slot. However, in the second schedule, slots are sorted according to the same sequence as nodes appear in the route from $A$ to $F$. This example shows that for a given route, slot misordering may induce delays. Of course, in multi-hop networks, this misordering may appear at each hop resulting in increased delays.

In this work we focus on delay optimization of the scheduling done by a coloring algorithm called VCM [5,6]. VCM is dedicated to grid networks and provides a periodic coloring obtained by repeating a specific color pattern. Our work takes advantage from the cross layering. Starting from a colored grid, we first build hierarchical routes taking into account the colors of nodes. Then, colors and hence slots granted to nodes on these routes are ordered following the appearance order of nodes on these routes.

The organization of this report is as follows. Section 3 presents the background of this work. In Section 4, we define the assumptions and the problem statement. Section 5 gives an overview of the solution and its main principles. Sections 6, 7, 8 and 9 describe the proposed solution. Section 10 concludes the report.

# 3 Background

## 3.1 Delay Efficient TDMA

Most delay-aware TDMA scheduling solutions for WSNs attempt to minimize the schedule length [1,2]. Other works rather focus on delays induced by the misordering of slots. In general, this latter category assumes that routes are already known and try to optimize the scheduling accordingly [9–13]. In [9], authors present an integer linear programming (ILP) formulation of the minimum latency link scheduling problem. Assuming that routes are given, the model is able to find the slot used by each link while minimizing the latency. We believe that the linear programming approach is not suitable for large networks. Solutions [11–13] assume that a data gathering tree is given. They ensure that any node has a slot earlier than its parent in the routing tree. Hence only one cycle is required for data aggregation. For instance, DMAC [12] allocates slots to nodes based on their positions in the data gathering tree. Nodes at the deepest levels in the tree are assigned the earliest slots. In SERENA [13] which is a scheduling algorithm based on graph coloring, each node has a color higher than the color of its parent in the data gathering tree. Consequently, when slots are assigned according to the decreasing order of colors, any parent node has a slot that follows the slots of all its children. We have shown in [13] that the cost of such ordering is the increase of the cycle length. Unlike DMAC, SERENA is conflict free while ensuring the spatial reuse.

Unlike previous works that assume that the routes are known and determine the minimum delay scheduling, Yu et al. [14] assume that the scheduling is known and determine routes that minimize delays. Indeed, each link $i, j$ is assigned a cost that is equal to the time difference between slots assigned to nodes $i$ and $j$ respectively. The advantage of this solution is that all routes (from any node to another) are possible without scheduling change, but on the other hand nothing makes them energy-efficient.

## 3.2 VCM: Vector-based Coloring Method

VCM [5, 6] is a graph coloring algorithm dedicated to dense grid networks. Indeed, the grid topology is one of the best methods to ensure full area coverage in surveillance applications [3, 4]. From the research point of view, working on grids can be a first step towards general graphs as it is easier especially for theoretical studies.
The intuitive idea of VCM is as follows. As the grid topology presents a regularity in terms of node positions, VCM produces a similar regularity

in terms of colors and generates a color pattern that can be periodically reproduced to color the whole grid. An example of the coloring provided by VCM is given in Figure 3.

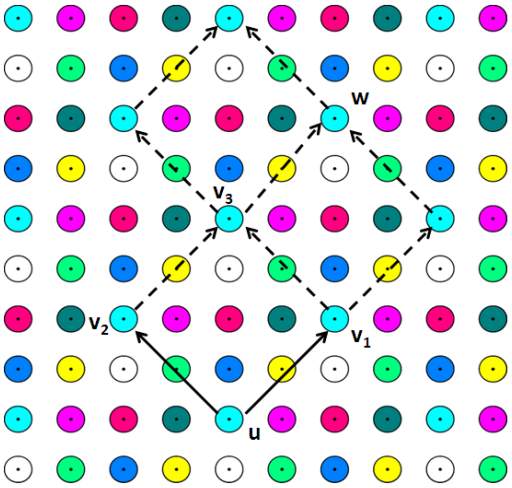



Figure 3: An example of 3-hop coloring provided by VCM (no 1 or 2 or 3-hop neighbors share the same color.)

The color pattern is defined by two vectors ($u\vec{v}_1$ and $u\vec{v}_2$ in Figure 3) called *VCM generator vectors*. VCM finds a color pattern that minimizes the number of colors used and hence minimizes the length of the TDMA cycle. VCM coloring has the following properties:

1. Nodes having the same colors form a lattice of generator vectors $u\vec{v}_1$ and $u\vec{v}_2$. Moreover, nodes with the same color are nodes with the same relative coordinates according to the parallelogram they belong to. Consequently, color computation is based on the coordinates of nodes relatively to their parallelogram.

2. All colors produced by VCM are the colors inside one parallelogram: The number of colors produced by VCM to color the whole grid for any transmission range $R$ is equal to the scalar product of the two generator vectors.

3. The number of colors produced by VCM does not depend on the size of the grid, but on the transmission range of sensor nodes.

In this work, we adopt this algorithm because: (1) the schedule it provides is optimal in terms of length when the transmission range tends to infinity, (2) nodes compute their colors using a simple formula without requiring message exchange with neighbors.

# 4 Assumptions and Problem Statement

## 4.1 Assumptions

- **Network and application**: We consider a grid wireless sensor network composed of $n$ sensors including a sink node. The nodes are disposed in a grid. Nodes are colored via VCM.

- **Application**: We assume a data gathering application where sensors monitor a region and transmit periodic measures or alarms to the sink. We assume also that aggregation is possible: any sensor is able to aggregate data it receives with the data it generates. This assumption is realistic especially when sensor nodes perform some operations like computing the average value or the maximum value of the received data.

- **Time slots**: We assume that each node is assigned a single time slot that suffices for all its transmissions.

- **The neighborhood:** The underlying assumption is that any two nodes $N$ and $M$ can communicate if and only if their euclidean distance is lower than the radio range $R$. Furthermore, the communications are assumed to be bidirectional.

## 4.2 Problem Statement

When the scheduling is based on colors, ordering the slots of a given schedule is equivalent to associating each color with a time slot in the right order (cf. Figure 1).

*The problem statement: Given a grid colored by the periodic coloring VCM, we want to build routes taking into account these colors and then to order these colors in a schedule ensuring that data reach the sink in minimized delays.*

We will see in the next sections that solving this problem is achieved by a cross layering between the scheduling and the routing.

# 5 Principles and Overview about the Solution

## 5.1 Discussion about Intuitive Solutions

Before describing the proposed solution, we explain the different steps that lead us to find the solution we propose under the aforementioned assumptions.

### 5.1.1 First solution: OPERA [7]

We have previously proposed the OPERA software composed of the routing EOLSR [7] and the scheduling OSERENA [8] adapted to random topologies. EOLSR builds a routing spanning tree and OSERENA colors the tree such that the color of any node is higher than the color of its parent. Colors are ordered in the cycle according to the decreasing order. Consequently, this coloring constraint allows the data aggregation in one cycle. However, for dense and large grid networks, coloring may produce many colors and VCM is more optimized than OSERENA in this case. Moreover, applying a routing algorithm that relies on a spanning tree like EOLSR and integrating it with a periodic coloring raises some issues as explained in the next section.

### 5.1.2 Second solution: Spanning Tree with VCM

This solution consists in building a spanning tree of the colored grid.

1. Assume first that we keep a random order of VCM colors on the TDMA cycle. The random order of colors in the TDMA cycle might result in slot inversion and hence additional delays as explained in the scenario of Figure 2. Hence, what we should do is to order colors of nodes on the routes.

2. Having a spanning tree that contains all nodes raises the following constraints:

   (a) Due to the periodicity of VCM colors, it is impossible that any tree branch contains any color ju +st once for large grids. In this case, delays might exceed one cycle.

   (b) Let a node $N$ and the route of this node towards the sink via the tree. We order the colors of intermediate nodes in this route such that the color of the first node that appears first in this route is associated to the first slot. This ensures one TDMA cycle for

data transmission for this node $N$. However, we cannot do this for all nodes, for two reasons. First, colors are shared between routes (color repetitions with VCM). Second, nodes are located at different design considerations of the data sink. So, optimizing the order of these colors for some nodes according to one direction might make this order not suitable for other directions.

The previous constraints lead us to define the following principles. Instead of building a unique spanning tree including all nodes where it is impossible to have delay optimal routes towards the sink for all nodes, we build several trees. Roots of these trees are particular nodes called aggregators. Then, we order the colors of nodes on these trees. Furthermore, each color does not appear more than once in any tree. Any aggregator routes data to another aggregator until data is received by the data sink. In the following two sections, a detailed description of the proposed solution is given.

## 5.2 Principles

1. A hierarchical routing structure is proposed: nodes transmit data to a set of aggregators and aggregators route data to the sink (cf. Figure 6).

2. Routing and scheduling must ensure that:

   - Any node reaches the closest aggregator in a single cycle. The path followed by this node is denoted '*Route*' in the remaining of this report.
   - The number of nodes that are able to reach more than one aggregator in a single cycle must be maximized. Hence, if a node cannot reach the closest aggregator because of link failure, it can route over another aggregator to reach the data sink. Also, this is useful also when there are multiple sinks. In this case, any node can select the closest aggregator for every sink.
   - Paths between aggregators should be fast. They are denoted *'Highways'* in the remaining of this report.

3. *Routes* and $Highways$ are activated successively in the TDMA cycle as depicted in Figure 4. The first activity period of the TDMA cycle is dedicated to routes: activation of VCM colors. Each VCM color occurs only once. The second period is a specific period dedicated to the activation of one $Highway$ (like the period $H1$ in Figure 4). Notice that the number of scheduled colors in this period is smaller than in

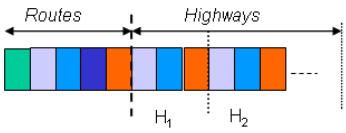



Figure 4: Orchestration of *Routes* and *Highways* slots in the TDMA cycle.

the first one. Unlike the first period, the second period is repeated a number of times sufficient to allow data sent by aggregators to reach the sink ($H1$ and $H2$ in Figure 4).

4. The scheduling is based on VCM colors. More precisely, no new colors are used and the spatial reuse obtained by VCM must still be guaranteed. Furthermore, these colors must be ordered in the TDMA cycle for both *Routes* and *Highways* periods, in such a way that the end-to-end delays are minimized.

5. Any node is active only during the slots associated with its color and the colors of its transmitters and can sleep the remaining time, both in the *Routes* period and the *Highways* period.

## 5.3 Overview

This section provides a more detailed description of the proposed solution. Figure 5 illustrates its different steps and components.

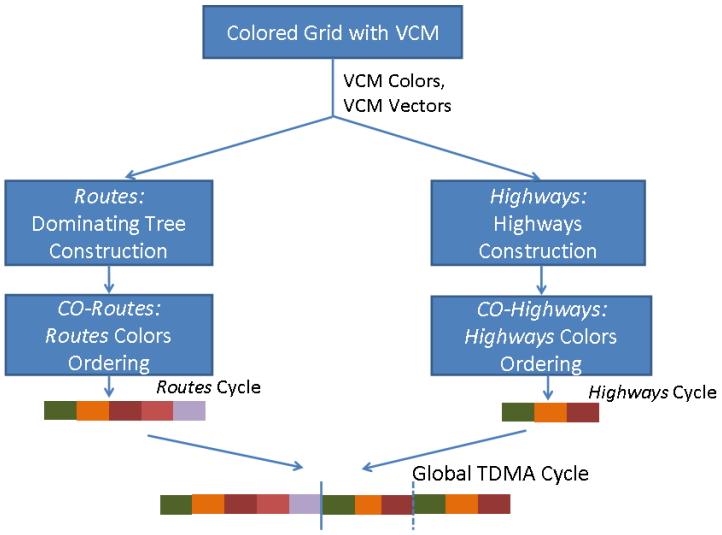



Figure 5: Steps of the proposed method.

First, the solution is applied to a colored grid of sensors where each node has a color computed by VCM. The principles of the solution can be presented in the following components:

1. **Hierarchical routing structure**: Figure 6 illustrates the hierarchical routing proposed. It is composed of *Routes* (dominating trees) and *Highways*. $u_1$ and $u_2$ are the VCM generator vectors.

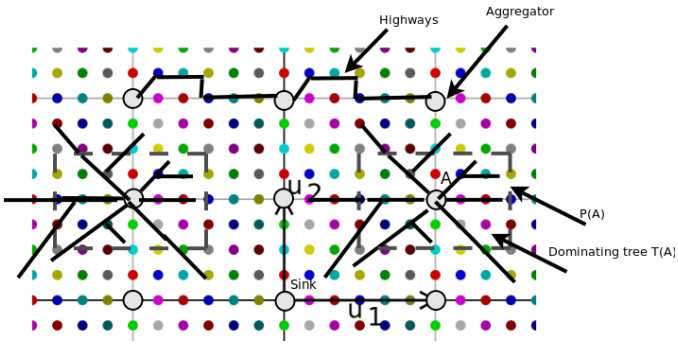



Figure 6: Hierarchical routing structure.

   (a) ***Routes***:

   - A set of aggregators is chosen: the vertices of VCM patterns (see Figure 6 as an example).
   - Each aggregator $A$ is the root of a dominating tree $\mathcal{T}_A$. A dominating tree is defined by Definition 1.

     **Definition 1.** *A tree $\mathcal{T}$ dominates a set of nodes if any node of this set is either in the tree or has at least one neighbor belonging to this tree. Such a neighbor is called dominator.*

     In other words, we build a connected dominated set (CDS) constructed as a tree which is rooted at the aggregator. The tree rooted at any aggregator $A_2$ is a copy of the tree rooted at another aggregator $A_1$ after translation by the vector $\vec{A_1A_2}$.
   - For any aggregator $A$, any tree $\mathcal{T}_A$ is as large as possible as long as any color appears only once in this tree branch.
   - Any node in the grid is necessarily dominated by the tree rooted at the closest aggregator and possibly by trees rooted at other aggregators. As previously said, these other trees are copies of the tree rooted at the closest aggregator after being translated.

(b) ***Highways***: Are used by aggregators to join the data sink. Except aggregators in the border of the grid, any other aggregator has 4 neighboring aggregators. We build 4 shortest paths between this aggregator and each of these 4 neighboring aggregators. These shortest paths are called *Highways*.

**Remark 1.** *Routes and Highways construction takes into account the colors of the nodes and is based on the same algorithm for all nodes. Hence, the obtained trees and Highways are the same for all aggregators (as illustrated in Figure 6). For this reason, in the following, we only reason about one aggregator.*

(c) **Routing**: Based on the aforementioned described architecture:

- Any node routes data over the dominating tree of the closest aggregator or another aggregator. Its next hop is: (1) either its parent in this tree if its belongs to the tree, (2) or its dominator otherwise.
- Then, we use geographic routing to determine for each aggregator the following aggregator over *Highways*.

2. **Ordering colors**: Recall that the slot assignment is based on colors computed by VCM. To ensure minimized delays avoiding slot misordering, we order the colors of nodes according to the appearance order of these nodes in the followed paths. In the remaining of this report, we speak about *colors ordering* defined as:

**Definition 2** (Colors Ordering)**.** *Given a set of colored nodes on a given path, each node color is associated with a time slot in such a way that the color of the node that appears first in the path (the farthest node from the destination) is associated with the first slot.*

To guarantee minimized delays over *Routes* and *Highways*, we perform a *Colors Ordering* as follows:

(a) ***CO-Routes (Colors Ordering of** Routes)*: The color of the first node linked to the tree is associated with slot 1, the color of the second node is associated with color 2, etc. Then, colors of remaining nodes are associated with remaining slots. This method ensures that the slot granted to a node increases with the node level in the tree. Also, any dominated node has a slot after its *dominator*.

(b) ***CO-Highways (Colors Ordering of*** *Highways)*: Similarly, for *Highways* the order of colors is the same as the order of the nodes on the routes from one aggregator to the next one.

3. **Orchestration**: The global TDMA cycle is composed of the *Routes* cycle and the *Highways* cycle. Notice that the *Highways* cycle is smaller. The *Highways* cycle itself is composed of sub-cycles repeated a number of times sufficient to allow the farthest aggregator to reach the data sink.

# 6 *Routes*: Dominating Tree Construction

Aggregators are nodes with coordinates $\alpha u + \beta v$ where $u$ and $v$ are VCM generator vectors and $\alpha$ and $\beta \in \mathbb{Z}$. Each aggregator is the root of a dominating tree. We define the problem of ***Dominating Tree Construction*** denoted ***DTC*** as:

***DTC Problem****: Finding a dominating tree with:*

- *The objectives:*
  - ***O1.*** *Each node dominated by a tree must reach the root of this tree in one cycle.*
  - ***O2.*** *The number of aggregators that any node can reach in a single cycle is maximized.*
  - ***O3.*** *Any tree dominates the largest possible number of nodes in the grid.*
- *The constraints:*
  - ***C1.*** *Each color appears only once in any branch of the dominating tree.*
  - ***C2.*** *Between any two branches of the tree, any two colors appear always in the same order when going on a path towards the root.*

In this section, we describe our algorithm to answer this problem. We start by introducing the following notation where $u$ and $v$ are VCM generator vectors.

**Notation 1** (Parallelogram $\mathcal{P}_A$)**.** *By abuse of notation, for a vector $w$ of coordinates $x_w, y_w$, we denote $\frac{w}{2}$ the vector of coordinates $\lfloor \frac{x_w}{2} \rfloor$ and $\lfloor \frac{y_w}{2} \rfloor$*[1]*.*

[1] $\lfloor . \rfloor$ is the symbol for integer part.

*Then, we define for any aggregator $A$ of coordinates $(0,0)$ the parallelogram $\mathcal{P}_A$ with vertices $(0, u, v, u+v)$ translated by $-(\frac{u}{2}+\frac{v}{2})$.*

## 6.1 Principles of the Construction of a Dominating Tree

We consider the aggregator with coordinates $(0,0)$. The obtained tree will be repeated for all other aggregators by virtue of the spatial repetition of VCM colors. We propose a connected dominating tree construction algorithm based on a "greedy" heuristic, as follows:

1. The tree is built in 2 steps: for the aggregator $A$ of coordinates $(0,0)$, we first build a connected dominating tree in $\mathcal{P}_A$ only. Then, we extend the obtained tree via the domination of nodes outside this area. The same algorithm is used for the interior and the exterior of this area.

2. The first node added to the tree is its root. All its neighbors are then dominated.

3. More generally, when any node is added to the tree, all its undominated neighbors are marked as dominated. The algorithm will not attempt to dominate them later.

4. Nodes are dominated in the order of their increasing number of hops towards the root. For two nodes that are at the same number of hops from the root, the node with the smallest geographic distance to the root is dominated first.

5. Nodes belonging to $\mathcal{P}_A$ must have dominators inside $\mathcal{P}_A$.

6. Nodes having VCM colors already used in the tree (color is identical to the color of a node already in the tree) cannot be dominated and cannot be used as dominators.

7. When any node $N$ is linked to the tree after having dominated one node, other nodes that have the same color and that are already dominated, are also linked to the tree.

8. Selection of dominators is based on a priority denoted *priority*. To define this priority, we use a heuristic. The priority of any node is the total number of the undominated neighbors that will be dominated if this node is added to the tree (with all other candidates of the same color).

## 6.2 Algorithm of the Construction of a Dominating Tree

Given the aggregator $A$ of coordinates $(0, 0)$, Algorithms 1, 2 and 3 illustrate the construction of the dominating tree $\mathcal{T}_A$ rooted at $A$.

**Algorithm 1** DOMINATING-TREE-CONSTRUCTION

Input: the grid $G$, the aggregator $A$ and the *priority* (the heuristic for the selection of the *dominator*)
Output: the dominating tree $\mathcal{T}$
$dominatedNodes = \{A\}$
$treeColors = \{color\ of\ A\}$
Add all the neighbors of $A$ to $dominatedNodes$
$insideNodes$ = list of nodes in $\mathcal{P}_A$ sorted according to the increasing distance in number of hops relative to the root of the tree. In case of equality, the closest node to the aggregator is the node with the smallest geographic distance to the sink.
$inside = True$
/*Try to dominate nodes inside $\mathcal{P}_A$*/
**for all** ($toBeDominated \in insideNodes$) **do**
    $dominator$ = DOMINATE-NODE ($\mathcal{T}$, $toBeDominated$, $inside$, $dominatedNodes$, $treeColors$, $priority$)
    **if** ($dominator \neq NULL$) **then**
        ADD-NODES ($dominator$, $\mathcal{T}$, $dominatedNodes$, $treeColors$) /*Adding dominated nodes with the same color as $dominator$*/
    **end if**
**end for**
$outsideNodes$ = list of nodes $\notin \mathcal{P}_A$ sorted according to the increasing distance in number of hops relative to the root of the tree. In case of equality, the closest node to the aggregator is the node with the smallest geographic distance to the sink.
$inside = False$
/*Try to dominate nodes outside $\mathcal{P}_A$*/
**for all** ($toBeDominated \in outsideNodes$) **do**
    $dominator$ = DOMINATE-NODE ($\mathcal{T}$, $toBeDominated$, $inside$, $dominatedNodes$, $treeColors$, $priority$)
    **if** ($dominator \neq NULL$) **then**
        ADD-NODES ($dominator$, $\mathcal{T}$, $dominatedNodes$, $treeColors$) /*Adding dominated nodes with the same color as $dominator$*/
    **end if**
**end for**

## 6.3 Properties of a Dominating Tree

In this section, we present some results concerning the dominating tree.

### 6.3.1 General Properties

**Lemma 1.** *Algorithm 1 generates a tree.*

*Proof.* When any node is dominated, a new link is added to the tree between this node and its dominator. Consequently, there is no disconnected parts in the grid. Hence the result. □

**Algorithm 2** DOMINATE-NODE ($\mathcal{T}$, *toBeDominated*, *inside*, *dominatedNodes*, *treeColors*, *priority*)

**if** (*toBeDominated* $\in \mathcal{T}$) OR (*toBeDominated* $\in$ *dominatedNodes*) OR (color of *toBeDominated* $\in$ *treeColors*) **then**
    return NULL /*This node cannot be dominated*/
**end if**
/*Try to dominate node *toBeDominated* */
**for all** (*neighbor* $\in$ neighbors of node *toBeDominated*) **do**
    **if** (*inside* == *True* and *neighbor* $\notin \mathcal{P}_A$) **then**
        continue /*next neighbor*/
    **end if**
    **if** ((*neighbor* $\notin \mathcal{T}$) OR (*neighbor* $\notin$ *dominatedNodes*) OR (color of *neighbor* $\in$ *treeColors*)) **then**
        continue /*next neighbor*/
    **else**
        Add *neighbor* to *possibleDominators*
    **end if**
**end for**
**if** (*possibleDominators* is empty) **then**
    return NULL /*Cannot dominate this node*/
**end if**
Sort *possibleDominators* according to *priority*
*dominator* = first node sorted in the list *possibleDominators*
Add *dominator* to $\mathcal{T}$
The parent of *dominator* = the dominator of *dominator*
Add color of *dominator* to *treeColors*
The dominator of *toBeDominated* = *dominator*
return *dominator*

**Algorithm 3** ADD-NODES (*dominator*, $\mathcal{T}$, *dominatedNodes*, *treeColors*)

**for all** (*node* $\in$ the grid $G$ with the same color as *dominator*) **do**
    **if** (*node* $\in$ *dominatedNodes*) **then**
        hasDominated = False
        **for all** (*neigh* neighbor of *node*) **do**
            **if** (*neigh* $\notin$ *dominatedNodes*) and (color of *neigh* $\notin$ *treeColors*) **then**
                hasDominated = True
                Add *neigh* to *dominatedNodes*
                The dominator of *neigh* = *node*
            **end if**
        **end for**
    **end if**
    **if** (hasDominated == True) **then**
        Add *node* to $\mathcal{T}$
        The parent of *node* = the dominator of *node*
    **end if**
**end for**

**Lemma 2.** *At each iteration, the closest undominated node $N$ to the root $A$ has necessarily at least one neighbor that is dominated by $\mathcal{T}_A$.*

*Proof.* Assume by contradiction that $N$ has no dominated neighbors. Since nodes form a grid (a connected graph), this assumption means that there exists a shortest path $N, N_1, \ldots, N_k, A$ between $N$ and the aggregator $A$, where $k$ is a positive integer. Two cases are possible:

- either $N_1$ is dominated, then there is a contradiction because we could select it as a dominator for $N$;
- either $N_1$ is not dominated and it is then the closest node to the root and not $N$ as assumed. This results in a contradiction. Hence the proof.

□

**Lemma 3.** *At any iteration, the algorithm attempts to dominate the closest node $N$ to the root aggregator that is not dominated. Two cases are possible,*

- *Case 1: $N$ cannot be dominated because:*
  - *its color is used by another node already in the tree;*
  - *all of its dominated neighbors have colors already in the tree;*
- *Case 2: $N$ can be dominated and a possible dominator exists.*

*Proof.* First, at any iteration, Algorithm 1 considers the closest node $N$ to the root and attempts to dominate this node if and only if this node is not dominated and this node has not a color used by a node belonging to the tree (see lines 1 and 2 in Algorithm 2). Second, from Lemma 2, $N$ has at least one neighbor that is dominated. If this neighbor has a color that appears in the tree, it is not kept (see line 10 in Algorithm 2). Otherwise, it is a possible dominator for $N$ (see line 12 in Algorithm 2). Hence the Lemma. □

**Theorem 1.** *Algorithm 1 meets the constraints of DTC problem.*

*Proof.* According to Algorithms 2 and 3, any color is linked to the tree in two occasions:

- First, to dominate any node, its dominator is added to the tree only if its color does not already appear in the tree (see lines 9 and 20 in Algorithm 2). Hence, no color repetition is produced. Let *dominator* denotes this dominator.

- Second, once *dominator* is linked to the tree, other nodes sharing the same color are added to the tree at the same time if they are already dominated and if they have undominated neighbors (see line 13 in Algorithm 3). This is the only time when nodes with this color are added to the tree. Because the tree is "grown" from the root, the same color cannot be repeated twice in the branch.

Now, to prove that Constraint **C2.** is met, let us assume that two colors $c_1$ and $c_2$ are on the same branch. Assume that $c_1$ is closer to the root than $c_2$. This means that all nodes with color $c_1$ had been added in the tree before all nodes with the color $c_2$ and are all closer to the root. Hence, the ordering of colors on the branches must be identical. □

**Theorem 2.** *A dominating tree addressing the DTC problem cannot dominate all nodes for all sizes of the grids.*

*Proof.* From Lemma 3, dominating one node becomes not possible when the color unicity of the tree might be not guaranteed. However, the number of colors is finite. Consequently, to guarantee the color unicity in the tree, some nodes might not be dominated. □

### 6.3.2 Properties Concerning Nodes in $\mathcal{P}_A$

**Lemma 4.** *Nodes in $\mathcal{P}_\mathcal{A}$ have all different colors.*

*Proof.* $\mathcal{P}_A$ is nothing other than the VCM pattern translated by the vector $\frac{u}{2} + \frac{v}{2}$. By definition, inside VCM pattern, there is no color reuse. □

**Theorem 3.** *Any node among $\mathcal{P}_\mathcal{A}$ is necessarily dominated by the dominating tree rooted at $A$.*

*Proof.* By contradiction, let us assume that there is one or several nodes in $\mathcal{P}_A$ that are not dominated. Let $M$ be the undominated node that is the closest to the aggregator. According to Lemma 2, node $M$ has at least one neighbor $D$ that is dominated. Because all nodes in $\mathcal{P}_\mathcal{A}$ (hence in the dominating tree $\mathcal{T}_A$) have different colors (Lemma 4), $D$ could have been selected as a dominator for $M$. Hence the contradiction. □

**Corollary 1.** *Any grid node is necessarily dominated by the tree rooted at the closest aggregator and possibly by trees rooted at other aggregators.*

*Proof.* The result is true because of Theorem 3 and because by construction, an aggregator $A$ is the closest aggregator to nodes in $\mathcal{P}_\mathcal{A}$. In addition, a tree rooted at $A_i$ can dominate nodes belonging to some $\mathcal{P}_{A_j}$ for $i$ and $j$ different positive integers. □

## 6.4 Results: Examples of Dominating Trees

Figure 7 depicts the dominating tree rooted at the aggregator of coordinates $(0, 0)$. The radio range is set to 3×the grid step. Thick lines present the tree links while the thin lines are the links between a node and the nodes it dominates. We observe that the obtained tree dominates a larger area: almost 16 color patterns.

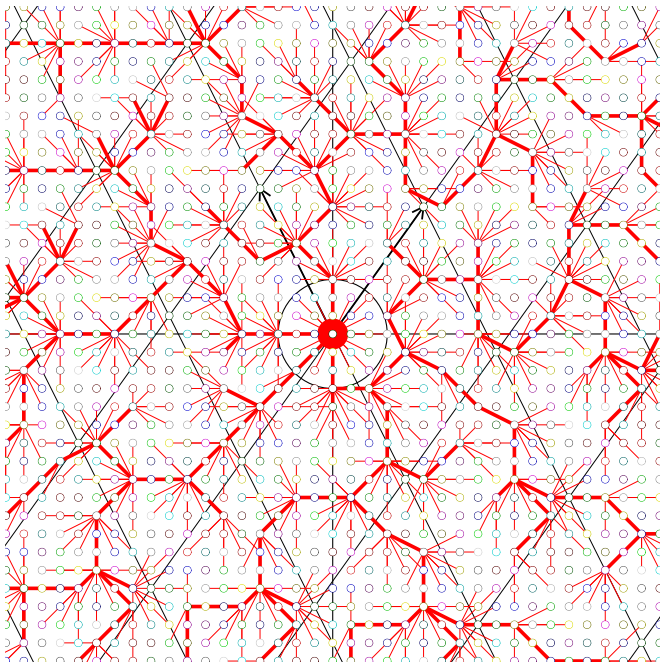


Figure 7: The dominating tree obtained for 3 hop coloring and radio range $= 3\times$ the grid step.

Figure 8 depicts two trees rooted at aggregators of coordinates $(0, 0)$ and $(6, -1)$. Only tree links are presented.

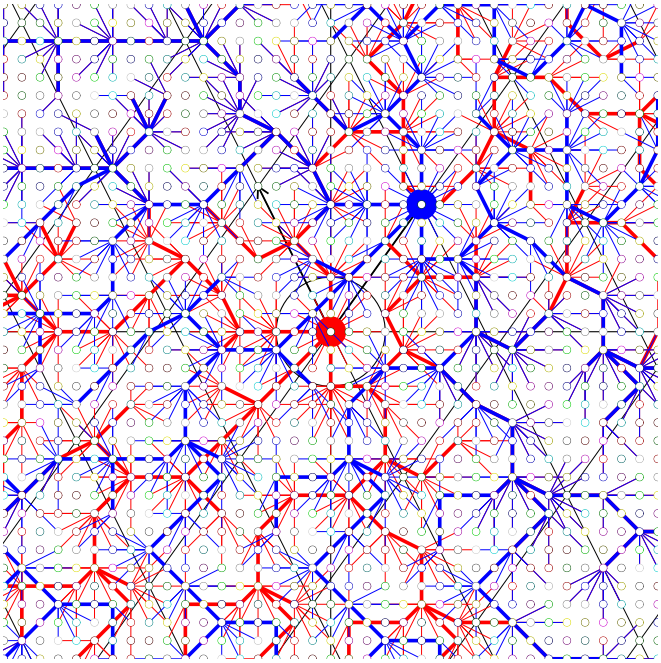


Figure 8: Two dominating trees for 3 hop coloring and radio range = 3× the grid step.

### 6.5 Routing Over the Dominating Tree

Any node that is dominated by a tree, including the tree rooted at the closest aggregator, can reach the root of this tree. Its next hop to reach this aggregator is:

- Either its parent in the tree, if it belongs to the tree.
- Or its dominator relative to this tree, otherwise.

## 7 CO-Routes: Colors Ordering for *Routes*

Our objective is to have *Colors Ordering* of *Routes*.

### 7.1 Method of CO-Routes

**Method 1** (CO-Routes)**.** *The method is based on the following steps:*

1. *Step1: Nodes of a tree:*
   *Consider the tree* $\mathcal{T}_A$ *rooted at the aggregator* $A$ *of coordinates* $(0, 0)$.

*We first start by ordering colors of nodes belonging to $\mathcal{T}_A$. During the construction procedure of the dominating tree (line 20 in Algorithm 2), the color of the first node added to the tree is associated with slot 1, the second (node or set of nodes) is associated with slot 2, etc. With this method, nodes with deepest levels in the tree are scheduled in the latest slots of the schedule.*

2. *Step2: Remaining nodes:*
   *The remaining grid nodes are sorted according to the increasing number of hops to the root of $\mathcal{T}_A$ and in case of equality the geographic distance is considered as a second criteria. Then, the colors of these nodes are associated with time slots starting with the last slot of the first step.*

**Remark 2.** *Method 1 produces a periodic coloring obtained by repeating $\mathcal{P}_A$ for any aggregator A. Consequently, no new colors than those of VCM are used and no color conflict are created.*

**Lemma 5.** *According to Method 1, for any aggregator A, colors of nodes inside $\mathcal{P}_A$ are ordered before nodes outside.*

*Proof.* Colors of nodes linked first to the tree are ordered firstly. Since the dominating tree construction starts by dominating nodes inside $\mathcal{P}_A$, we have the result. □

## 7.2 Results about the Delays Obtained

**Lemma 6.** *Nodes belonging to any tree $\mathcal{T}_A$ reach A in one cycle.*

*Proof.* Let $N$ be any node in $\mathcal{T}_A$. By Method 1, $N$ will be scheduled after all its ascendants since they are linked to the tree before node $N$. Moreover, from Theorem 1, no two nodes have the same color among these ascendants. Because of Constraint C2., the ordering of colors is consistent on all branches. Hence, no slot misordering is produced and no more than one cycle is needed by any node $N$ to reach this aggregator $A$. □

**Lemma 7.** *All nodes that reach any aggregator A in one cycle are necessarily dominated by the tree $\mathcal{T}_A$.*

*Proof.* Assume that a node can reach the aggregator in one cycle. This means that there is a path to the root with only increasing slots numbers. Algorithm 1 adds nodes to the tree one after the other, and orders their colors at the same time. Iteratively, each node of the path, in the reverse order, will be successively dominated by the next-hop and later linked to the tree. Therefore, the path will be (part of) a branch of the tree. □

**Lemma 8.** *For any aggregator $A$, for any node $N \in \mathcal{P}_{\mathcal{A}}$, $N \notin \mathcal{T}_A$ and $N \in \mathcal{T}_{A_1}$ where $A_1$ is another aggregator, then $N$ is scheduled after its dominator relative to the tree rooted at $A$.*

*Proof.* According to Lemma 5, the colors of nodes inside $\mathcal{P}_{\mathcal{A}}$ is ordered before the part outside. Thus, the color of $N$ is ordered after all nodes in $\mathcal{P}_{A_1}$. This means that the color of $N$ is ordered after all nodes in $\mathcal{P}_A$ (because $\mathcal{T}_A$ is a copy of $\mathcal{T}_{A_1}$). Hence, $N$ has a slot after its dominator relatively to $\mathcal{T}_A$.

□

**Theorem 4.** *Given any aggregator $A$, nodes in $\mathcal{P}_{\mathcal{A}}$ reach the aggregator $A$ in one cycle.*

*Proof.* Let $N$ be a node in $\mathcal{P}_{\mathcal{A}}$. From Lemma 3, this node has necessarily one dominator which is its next hop towards the aggregator $A$. Three cases are possible:

1. $N \in \mathcal{T}_A$: the result comes from Lemma 6.
2. $N \notin \mathcal{T}_A$ and does not belong to any tree. According to Method 1, the color of $N$ is ordered after nodes in the tree, and hence after its dominator. This dominator is able to reach the sink in a single cycle (see Lemma 6).
3. $N \notin \mathcal{T}_A$ but belongs to a tree $\mathcal{T}_{A_1}$ rooted at another aggregator $A_1$. According to Lemma 8, $N$ is scheduled after its dominator, itself able to reach the sink in a single cycle according to Lemma 6.

□

# 8 *Highways* Construction and Coloring

*Highways* are paths between aggregators and the data sink. Data are routed from one aggregator to another in the direction of the sink. These paths are called *Highways* because they are fast; they comprise a small number of nodes and slots granted to these nodes are ordered according to the appearance order of nodes in these paths. In this section, we detail these aspects.

## 8.1 *Highways* Construction and Routing over *Highways*

The construction of *Highways* is based on the following steps:

1. Considering the grid composed of aggregators (that is a grid where vertices are the aggregators), except aggregators on the grid border, other aggregators have 4 neighboring aggregators. That is, if we consider the aggregator $A_0$ placed at the center of the grid, this aggregator has 4 neighboring aggregators $A_u$, $A_v$, $A_{-u}, A_{-v}$ placed respectively at nodes of coordinates $u$, $v$, $-u$ and $-v$ where $u$ and $v$ are the VCM vectors.

2. We build 4 shortest paths between the aggregator $A_0$ and each of its neighboring aggregators $A_u$, $A_v$, $A_{-u}, A_{-v}$. For this reason, we speak about 4 *Highways*: $H_u$ on the direction of $u$ vector (between $A_0$ and $A_u$), the second one denoted $H_v$ in the direction of $v$ vector, the third one denoted $H_{-u}$ is in the direction of $-u$ vector and the fourth one denoted $H_{-v}$ in the direction of $-v$ vector.

3. Two different *Highways* can go through the same nodes.

4. The *Highways* built for the aggregator $A_0$ are repeated for all other aggregators. Notice that aggregators that are in the border of the grid do not need to use all of these *Highways*.

5. The previous steps allow each aggregator to have *Highways* in four directions. These *Highways* are used for routing as follows: We use a geographic routing like [15] to select the closest neighboring aggregator to the sink. The path followed to reach this aggregator is the *Highway* computed in step 2.

**Remark 3.** *Each aggregator has Highways in* 4 *directions. This ensures a routing flexibility especially in the following scenarios: the position of the sink is unknown beforehand, the sink is mobile or when there are multiple sinks. Furthermore, it increases robustness against routing failures.*

**Remark 4.** *Notice that building Highways for further directions means activating more nodes in H period. Hence, the number of built Highways for any aggregator is subject to a compromise between delays and energy consumed per node.*

## 8.2 CO-Highways: Colors Ordering for *Highways*

After building *Highways*, we now order the colors of nodes in these *Highways* in a specific part of the TDMA cycle dedicated to *Highways*. Indeed, for each *Highway*, we order the color of intermediate nodes of this *Highway* such that the color of the $i^{th}$ node on this path is associated with the time slot number $i$ in the part of the TDMA cycle dedicated to *Highways*.

**Theorem 5.** *With Highways, no more than one cycle is required to any aggregator to reach one of its neighboring aggregators.*

*Proof.* Nodes on *Highways* have colors that are sorted according to the order of nodes in *Highways*. So, each node is scheduled before its next hop. Hence the theorem. □

# 9 Orchestration in a TDMA cycle

The key idea of the proposed architecture is that each packet is transmitted first from its source node to the closest aggregator, then this packet is transmitted from one aggregator to another via *Highways* until it reaches the data sink. In this section, we describe how to carry out this orchestration.

Figure 9 depicts the global TDMA cycle obtained. This cycle is composed of two periods:

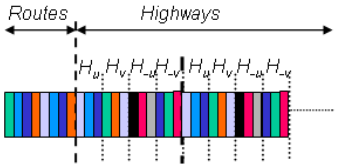



Figure 9: Global TDMA cycle.

1. First period denoted $R$ corresponding to the schedule of *Routes*. In this period, each node is awake during its slots and during the slots of its children in the dominating tree. Let $|R|$ be the number of slots of this period.

2. Second period denoted $H$ composed of colors of the *Highways*. This period is divided itself into 4 sub-periods: $H_u$, $H_v$, $H_{-u}$ and $H_{-v}$. Each period $H_w$ corresponds to a *Highway* between two aggregators optimized according to the direction of vector $w$ (see Section 8.1). The alternation of the 4 directions in $H$ cycle favors access fairness between aggregators that follow different directions to reach the sink. The period $H$ is repeated a number of times until the farthest aggregator from the sink reaches this sink. In this period, each node belonging to any *Highway* is awake during its slots and during the slots of nodes for which it is a next hop. Let $|H|$ denotes the number of slots of this period.

**Lemma 9.** *$H$ cycle is repeated a number of times equal to $max(|X-X_0|, |Y-Y_0|)$ times, where $X_0$ and $Y_0$ are the coordinates of the sink and $X$ and $Y$ are the coordinates of the farthest aggregator from the sink.*

*Proof.* $H$ cycle is repeated a number of times until all aggregators reach the sink. Hence, this number of times depends on the distance separating the farthest aggregator $A$ to the sink and this sink. Following the cycle depicted in Figure 9, in one cycle $H$, $X - X_0$ and $Y - Y_0$ are decremented by 1 unit. Hence, $A$ will reach the sink when the maximum between $|X - X_0|$ and $|Y - Y_0|$ is equal to *zero*. Hence, $max(|X - X_0|, |Y - Y_0|)$ cycles are required. □

# 10 Conclusion

This report addressed the TDMA delays induced by the misordering of time slots. We proposed a centralized solution jointly addressing routing and scheduling for data gathering applications in grid WSNs. We proposed a hierarchical routing based on a dominating tree construction. The idea is to enable each node to transmit data to the closest aggregator and then packets follow *Highways* from one aggregator to another.

The scheduling algorithm we consider is based on a previous work about grid coloring (VCM). Once trees are built, we order in the cycle the colors already computed by VCM to minimize data collection delays. This solution allows nodes to reach the closest aggregator in one cycle. Furthermore, *Highways* are very efficient as they comprise a small number of slots that are ordered according to the appearance order of nodes on these paths. As a future work, we plan to extend this work for general topologies.

# References


[1] Incel D.O., Ghosh A., Krishnamachari B., Chintalapudi K., *Fast data collection in treebased wireless sensor networks*, IEEE Transactions on Mobile Computing, 2009.

[2] Ergen S.C., Varaiya P., *TDMA scheduling algorithms for wireless sensor networks*, Wireless Networks, May 2010.

[3] Baumgartner T., Fekete S.P., Kamphans T., Kröller A., Pagel M., *Hallway Monitoring: Distributed Data Processing with Wireless Sensor Networks*, REALWSN, December 2010.

[4] Chakrabarty K.; Iyengar S. S.; Qi H. R. and Cho E., *Grid Coverage for Surveillance and Target Location in Distributed Sensor Networks*, IEEE Transactions on Computers, vol. 51, pp. 1448-1453, 2002.

[5] Adjih C., Amdouni I., Minet P., *VCM: the vector-based coloring method for grid wireless ad hoc and sensor networks*, MSWIM 2012, October 2012.

[6] Amdouni I., Adjih C., Minet P., *On the Coloring of Grid Wireless Sensor Networks: the Vector-Based Coloring Method*, Inria Research Report, October 2011.

[7] Website of OPERA software:
`opera.gforge.inria.fr/index.html`

[8] Amdouni, I.; Minet P.; Adjih C., *OSERENA: a Coloring Algorithm Optimized for Dense Wireless Networks*, The International Journal of Networked and Distributed Computing (IJNDC), vol. 1, pp. 9-24, IJNDC, January 2013.

[9] Cheng M.X., Gong X., Xu Y., Cai L., *Link activity scheduling for minimum end-to-end latency in multihop wireless sensor networks*, in IEEE Globecom'11, 2011.

[10] Cheng M.X.,Ye Q., Cai L., *Cross-Layer Schemes for reducing Delay in Multi-hop Wireless Networks*, IEEE Transactions on Wireless Communications, vol. 12, no. 2, pp. 928-937, February 2013.

[11] Chatterjee P., Da N., *A cross-layer distributed TDMA scheduling for data gathering with minimum latency in wireless sensor networks*, Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology, May 2009.

[12] Lu G., Krishnamachari B., Raghavendra C., *An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Wireless Sensor Networks*, Parallel and Distributed Processing Symposium, April 2004.

[13] Amdouni I., Minet P., Adjih C., *Adaptivity of a Coloring Algorithm to Unreliable Communications for Data Gathering in Wireless Sensor Networks*, International Journal of Digital Information and Wireless Communications (IJDIWC), vol. 3, pp. 61-74, Avril 2013.

[14] Yu F., Wu T., Biswas S., *Routing with Minimized Slot Misordering for Delay Mitigation in TDMA based Sensor Networks*, Third International Conference on Networking and Services (ICNS 2007), June 2007.

[15] Shaochuan S., Jiayan Z., Wenbin Z., *Grid-based Autonomous Geographic Routing in Wireless Sensor Networks*, IEEE International Conference on Computer Science and Automation Engineering (CSAE 2012), May 2012.