Routing-aware TDMA scheduling for Wireless Sensor Networks

Lemia Louail, Violeta Felea
FEMTO-ST Institute, University of Bourgogne Franche-Comté (UBFC), Besançon, France
$\{firstname.lastname\}$@femto-st.fr

***Abstract*—In Wireless Sensor Networks, TDMA schedulings are designed at the data link layer in the layered protocol stack to improve some metrics as the latency. Decisions of routing at the network layer are dissociated of the MAC communication planning, which cannot globally optimize latency. Cross-layer designs overcome this problem by ensuring communications between protocols of different layers. A TDMA scheduling can be more efficient in terms of latency when using information of the routing protocol.**

In this context, we present IDeg-LO and IDeg-ReLO, new TDMA schedulings using information of the routing tree. We define a new metric per node, the interference degree, involved in the node ordering used by the slot allocation process.

The simulation results show better performance of proposed scheduling algorithms compared to the state of the art. IDeg-LO improves average latency from 13.19% up to 44.14% depending on the routing tree, while IDeg-ReLO has better average latency improvements from 33.53% up to 53.33%.

I. Introduction

Wireless sensor networks enable connectivity between small devices, equipped with data processing and communication capabilities, intended to monitor physical or environmental conditions and to cooperatively transmit data to a central location commonly called sink. These devices integrate the necessary sensors, therefore they are referred to as sensor nodes. These nodes can easily cover large geographical areas, requiring multi-hop communications. The network architecture considered here is made of a single, fixed sink and fixed sensor nodes and the communication pattern is from nodes to sink.

A sensor node has a simplified OSI stack composed of the physical layer, the data link layer, the network layer, the transport layer and the application layer. The OSI stack plays an important role as independent functionalities can be implemented in each layer. More particularly, we are interested in two layers: the data link layer and the network layer, both involved in communication decisions. The MAC protocol at the data link layer is involved in coordinating communications between a node and its direct neighbors, while the routing protocol at the network layer is responsible of the route a packet should take to be transferred from a source to the sink. Both types of communication, one hop and multi-hop, generate latencies, dependent either on the MAC protocol or on the computed route. Generally, minimizing this latency in layered architectures is better achieved when using cross-layer approaches [1], [2]: information from one layer is used in decisions taken by protocols of another layer.

The scope of this paper is to determine suitable TDMA schedules for all nodes of a sensor network, correlated to the route computed by the routing protocol. The coordination between the data link and network layers makes of this solution a cross-layer approach. We consider only tree-based routing approaches, where each node has a single parent to forward its packets, and this, towards the sink. It is a common architecture for sensor network applications where a single sink collects data sensed by the nodes. We define a suitable schedule as the one which minimizes communication latency for every node when it sends packets to the sink. We define heuristics based on the idea of scheduling slots for nodes according to a particular node order. The interference degree is the metric which gives indication on this order.

The remainder of this paper is structured as follows. Section II presents similar existing works in the cross-layering where MAC scheduling decisions are based on the routing information at the network layer. In section III are presented our sensor network model and the representation of a TDMA schedule. Section IV shows a motivating example together with a similar cross-layer existing approach. Sections V and VI describe the heuristics we propose in order to define TDMA schedules for all nodes of the network, with the scope of minimizing concurrently all node-to-sink latencies. Section VII mentions the scenario description and metrics used to evaluate the proposed heuristics. Section VIII presents and analyzes results of simulations. The last section concludes.

II. Existing work

We cite two main classes of cross-layer approaches using communication scheduling based on routing information: distributed (MAC-CROSS, RMAC, CL-MAC, and AreaCast) and centralized (CoLaNet).

MAC-CROSS [3] and RMAC [4] both use the duty-cycle principle to reduce energy consumption of nodes which are not currently involved in communication. The main idea of MAC-CROSS is to maximize the sleep duration of sensor nodes and to minimize the number of nodes that are supposed to wake up. Whenever nodes are not involved in communication (not on the routing path), information given by the routing protocol at the network layer, their radio transceivers are turned off. RMAC is also a duty-cycle MAC protocol that exploits cross-layer routing information. Each intermediate relaying node for the data packet along these hops sleeps and intelligently wakes up at a scheduled time, so that its upstream node can send the data packet to it and it can immediately forward the data packet to its downstream node. RMAC can thus deliver a data packet much faster without sacrificing the energy efficiency achieved by the duty-cycle mechanism.

CL-MAC [4] is a cross-layer MAC protocol that uses routing information to transmit multiple data packets over multiple multi-hop flows. It considers all pending packets in the routing buffer and pending flow setup requests (from other nodes) in its contention-based channel allocation. In scheduling communication, it makes nodes later in the destination field wait longer based on the fact that packets of shorter paths suffer less end-to-end delay than those having longer paths to travel.

AreaCast[5] is a MAC-layer mechanism that uses local topological and routing information to provide a communication by area instead of traditional node-to-node communications and this by electing three implicit relay nodes within an area close to the explicit relay node.

CoLaNet[6] constructs a tree-based routing structure at the sink, based on the hypothesis of many-to-one applications traffic. Using it, the sink applies the channel assignment algorithm to decide the TDMA transmission schedule of each sensor node. To ensure a collision-free transmission and maximize the system capacity, CoLaNet first transforms this channel assignment problem for a sensor network into a vertex-coloring problem in its corresponding graph.

The CoLaNet assignment is dependent on both the sensor network and the routing tree, with the scope of contention-free scheduling. This objective does not cover the end-to-end communication delay. Transmission slots for the nodes in relation in the routing tree should be as close as possible, and this in the direction of communication given by the routing tree, from leaves to the root.

Our contribution tackles TDMA slot scheduling for nodes in order to optimize the end-to-end communication delay simultaneously for every sensor node communicating data to the sink. It is a centralized approach, which makes it comparable to the CoLaNet scheduling solution.

The main idea is to find an order for nodes to be scheduled and to determine a communication slot for every node, trying to minimize gap between every node and its direct predecessor given by the routing tree.

III. Sensor network and TDMA schedule modeling

Traditionally, a sensor network is modeled as a graph with a set of vertices representing the sensor nodes and a set of edges corresponding to the links between nodes. The way links are connecting nodes depends on the connectivity model. The classical connectivity model is based on the unit-disk graph [7] (UDG) where any two nodes are adjacent if and only if their Euclidian distance is at most 1. Applied to sensor networks, a node x has a wireless communication link with another node y if y is in within x's communication range. We consider in this study that all nodes have the same communication range, therefore the links in the modeled graph are symmetric. Moreover, links are considered reliable. An example of this kind of graph is given at the left side of Figure 1.

The routing protocols considered here construct single, fixed, multi-hop paths: every node, except for the sink, has a single forwarding node. Therefore, we model routing information as an oriented tree. In the routing tree every node, except for the sink, can be source of data and may generate packets to be transmitted to the sink via the routing path. As the sink in a sensor network is the destination of all collected data, it will always be the root of the routing tree. On the right side of Figure 1 is illustrated a routing tree for the sensor network given on its left, where node 1 is the sink node, therefore the root of the tree.

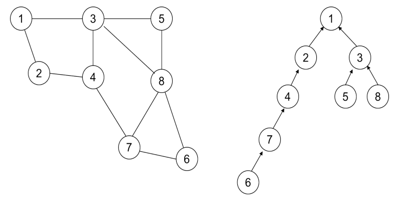
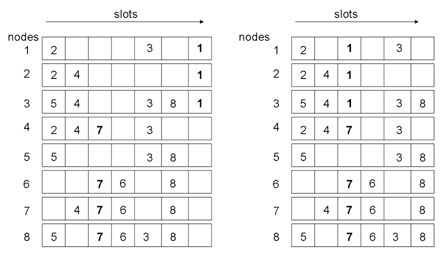


Fig. 1. A sensor network graph and one of its routing tree

TDMA uses the same frame length for every node. The frame is divided in slots, a slot being the lapse of time a node is in communication with one of its neighbors (either in reception or in transmission). The frame length is measured in number of slots. The minimum frame length corresponds to the maximum node degree incremented (a node is either transmitting or receiving data from any of its neighbors). The maximum frame length corresponds to the number of nodes in the network (the maximum degree of a node is n-1, where n is the number of nodes in the network).

Let us consider the frame length to be equal to k. A TDMA schedule for a node i is modeled by a vector $sched_i$, with $1 \leqslant i \leqslant k$, as follows: $sched_i^j = i$ if node i transmits at slot j, $sched_i^j = k$ if node i receives data from a direct neighbor k at slot j. The value $sched_i^j$ may not be defined if the node i is neither transmitting nor receiving at slot j.

Let us consider the following TDMA schedules (see Figure 2) for the nodes in the sensor network modeled by the graph in Figure 1. The schedule on the left has a frame length of 7 slots. For example, node 4 transmits at the second slot, while it receives from its neighbors 2, 3 and 7 at slots 1, 5 and 3 respectively.



nodes	slots						
1	2				3		**1**
2	2	4					**1**
3	5	4			3	8	**1**
4	2	4	**7**		3		
5	5				3	8	
6			**7**	6		8	
7		4	**7**	6		8	
8	5		**7**	6	3	8	

nodes	slots					
1	2		**1**		3	
2	2	4	**1**			
3	5	4	**1**		3	8
4	2	4	**7**		3	
5	5				3	8
6			**7**	6		8
7		4	**7**	6		8
8	5		**7**	6	3	8

Fig. 2. TDMA schedules for the sensor network in Figure 1

We can see that the transmission slot of node 1, the seventh one, can be scheduled at the same time with the transmission slot of node 7 (marked in bold) because no interference is generated. We obtain the TDMA schedule on the right in Figure 2, whose frame length is smaller (6 instead of 7). We note that the theoretical inferior bound of the frame length for this example (5 = maximum node degree + 1) does not provide collision free communications. Six slots are required for the frame length to ensure this property.

These schedules are obtained in an empirical way. The objective is to schedule communication such that no interference should be generated when a node transmits. This concerns nodes in immediate neighborhood of the nodes and nodes in its 2-hop neighborhood. For nodes in the 3-hop neighborhood of a node x, they can use, for transmission, the same slot as the transmission slot of node x. It is the case in the previous examples for nodes 2 and 5 which use the same slot, 1, because they are 3 hops away one from the other, so no interference can appear when they try to communicate simultaneously.

We denote in the followings, this empirical scheduling as *random*.

IV. TDMA-SCHEDULED, MULTI-HOP COMMUNICATION

When correlating TDMA schedule and multi-hop routing, frames are repeated such that every node has its communication slot (for transmission) every new frame. This may generate delay when data is transmitted. Let us consider the random slot schedule (see the right side of Figure 2) for the graph in Figure 1.

Consider data transmitted by node 6, leaf in the routing tree (see Figure 1). The first slot is considered the referential time of the sensor network running. Before node 6 can transmit, it has to wait its transmission slot, the forth. Transmission to its parent, node 7 in the tree, would take 1 slot. Afterwards, its parent forwards data further, towards the sink. Node 7 has to wait 4 slots before being able to transmit, at its turn, during the third slot (frames are repeated). Summing up, node 4 (forwarding node for node 7) will get data of node 6 after 3+1+4+1=9 slots for this TDMA schedule.

We describe briefly the TDMA schedule based on the routing tree proposed in [6]. Their approach, called CoLaNet, constructs the routing tree (the MinDegree tree) considering the sink as its root, the neighbors of the sink are its children and then each node chooses the node with the fewest children nodes as its parent. CoLaNet constructs a TDMA schedule using an approximation algorithm for the vertex-coloring problem [8]. When all nodes have a color, the colors are transformed into a schedule where the number of colors represents the length of the schedule, and each color represents the transmission slot for the node.

CoLaNet first colors the node with the Maximum Degree in the routing tree. Then, it continues to color vertices that have an already colored neighbor. If two or more vertices have a colored neighbor, no order is applied on them, they are taken randomly one by one. When giving a color to a node, CoLaNet ensures that none of its one-hop or two-hop neighbors in the graph has this color and this to avoid collisions. If not, a new color is generated and is given to the node.

The coloring algorithm takes two input parameters: the graph modeling the sensor network (a Graph object) and the routing tree giving the routing information (a RoutingTree object). The result is a Hash Table (a ColorTable object) containing each node with its corresponding color. The algorithm is summed up in the followings, where:

- `colorTable.coloring(n)` colors the node n while ensuring that none of its one-hop or two-hop neighbors in the graph already has this color, then saves the node with its color in the `colorTable`,
- `addLast` and `removeFirst` are list primitives,
- `ListOfNodes` is a List structure containing the nodes.

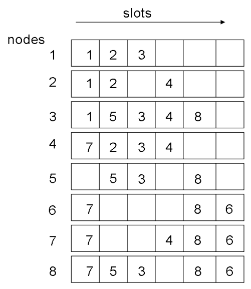
```
ColorTable<Node,Color> CoLaNet(
        Graph<Node> graph,
        RoutingTree<Node> routingTree){
  // graph: models the sensor network
  // routingTree: gives the routing information

  ColorTable<Node, Color> colorTable =
               new ColorTable<Node, Color >();

  ListOfNodes<Node> nextToColor =
                   new ListOfNodes<Node >();
  nextToColor.addLast(graph.getMaxDegreeNode());

  while(!nextToColor.isEmpty()){
    Node n = nextToColor.removeFirst();
    colorTable.coloring(n);
    for(Node node : neighbors of n)
      nextToColor.addLast(node);
  }
  return colorTable;
}
```

Let us consider the sensor network modeled by the graph in Figure 1 (on the left side) and its MinDegree routing tree given in the same Figure (on the right side). The CoLaNet schedule resulting from this tree is given in Figure 3.


slots
nodes

1	1	2	3			
2	1	2		4		
3	1	5	3	4	8	
4	7	2	3	4		
5		5	3		8	
6	7				8	6
7	7			4	8	6
8	7	5	3		8	6

Fig. 3. TDMA CoLaNet schedule for the sensor network in Figure 1

Using the CoLaNet schedule given previously and the same routing tree in Figure 1, data sensed by node 6 and transmitted towards the sink, will get to node 4 after 5+1+1=7 slots. However, when considering data transmitted from node 4, it will take 7 slots before it arrives at the sink, using the random scheduling, while the CoLaNet scheduling makes the data arrive after 8 slots.

Neither of the previous scheduling algorithms (random or CoLaNet) takes into account the routing tree to minimize data delivery time. Nevertheless, the data delivery time is also dependent on the routing tree. This is why we propose to integrate routing information into the scheduling algorithm in order to minimize delivery time for all nodes having data to send. Optimal scheduling can be obtained by enumerating all valid schedules, but the complexity of the algorithm is exponential, so only heuristics are reasonable in time. This is the scope of the next section.

V. INTERFERENCE DEGREE LEAVES ORDER - IDEG-LO

The slot scheduling may focus only on the transmission slots, one per every node. The reception slots may be inferred based on the graph representing the sensor network.

In order to schedule communications for nodes, we begin by scheduling the leaves in the routing tree. Compared to their predecessors, their paths are the longest, so sooner their slots are scheduled, less time they will wait before being able to communicate. Moreover, leaves are considered by decreasing order depending on their interference degree. The *interference degree* for a node is the sum of the number of one-hop neighbors and the number of two-hop neighbors counted only once (if a node is a one-hop neighbor and a two-hop neighbor at the same time, it is considered only once). Indeed, these nodes create interference when scheduling slots. Higher the interference degree, fewer are the options to schedule a slot for the node, without adding a new slot. This remark makes us considering leaves to be scheduled in the descending order of the interference degree. After that, internal nodes are scheduled in the same order as the leaves. Every node is scheduled as close as possible to its children's slots (within one hop distance) and after them, possibly cycling to the beginning of the schedule. The frame length is initialized to its lower bound. When searching for a compatible slot for a node, if no solution exists, a supplementary slot is added and the node is scheduled on this slot.

The algorithm, called IDeg-LO, is summed up in the followings, where Schedule<Node> models the schedule (the slots) of a node.

```
Schedule IDeg-LO(Graph<Node> graph,
           RoutingTree<Node> routingTree) {

  Schedule sched = new Schedule(
             graph.getMaxDegree()+1);

  ListOfNodes<Node> leaves =
           routingTree.getLeaves();
  leaves.sortByInterferenceDegree(graph);

  for (Node n : leaves)        // loop 1
    sched.findFreeSlot(n, graph, routingTree);

  ListOfNodes<Node> toSchedule =
                  new ListOfNodes<Node>();

  for (Node n : leaves) {     // loop 2
    Node parent = routingTree.findParent(n);
    if (!toSchedule.contains(parent))
      toSchedule.addLast(parent);
  }

  // Leaves are taken randomly in loop 1
  // and are taken in loop 2 in the same
  // order as in loop 1

  for (Node n : toSchedule) {
    Node parent = routingTree.findParent(n);
    if (!toSchedule.contains(parent))
      toSchedule.addLast(parent);
  }

  while (!toSchedule.isEmpty()) {
    Node next = toSchedule.removeFirst();
    sched.findFreeSlot(next, graph, routingTree);
  }
  return sched;
}
```

In the IDeg-Lo scheduling, the main algorithm which affects a slot to a node is `findFreeSlot`, applied to a schedule, which takes as input parameters: the node for which the slot is searched for, the graph and the routing tree.

The modeling of the schedule is done with a Schedule Class in which each node has a `Vector slots[]` representing its slots.

```
void findFreeSlot(Node node, Graph<Node> graph,
                RoutingTree<Node> routingTree){
   // method of the Schedule object which
   // determines the transmission slot
   // for the node 'node'

   boolean placeFound=false;
   if (node.isLeaf()) {
     for (int i=0; i<this.length(); i++) {
       if (slots[i] not taken by any of the
            one-hop or two-hop neighbors) {
         this.set(i, node);
         placeFound = true;
         break;
       }
     }
   } else { // the node is a parent
     // Find the slots of all its children
     // and use the max of them
     int max = this.findMaxSlotChildren(node,
                          graph, routingTree);
     int i = max+1;
     while (i != max) {
       if (slots[i] not taken by any of the
              one-hop or two-hop neighbors) {
         this.set(i, node);
         placeFound = true;
         break;
       }
       i = (i+1)%this.getLength();
     }
   }
   if (!placeFound) {
     this.addSlotAtEnd();
     this.set(this.getLength()-1, node);
   }
 }
```

We used an object-oriented paradigm to present algorithms, where `this` makes reference to the current object (here the Schedule object).

Let us consider the study case in Figure 4, based on a particular sensor network modeled by the graph at the left side of the figure and its routing tree in its right part.

The order of the nodes to be scheduled, according to the IDeg-LO algorithm, is the following: 5, 6, 2, 7, 4, 3, 8,

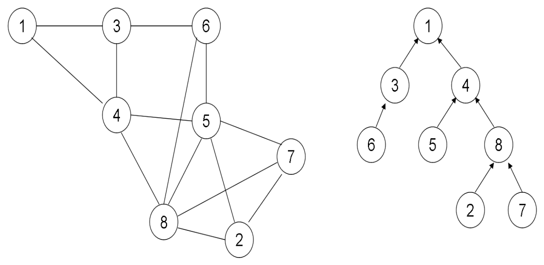


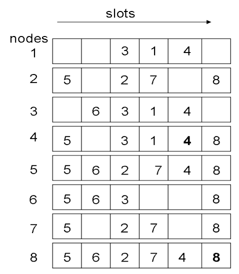
Fig. 4. A sensor network graph and its routing tree

1 (corresponding to the interference degree of the leaves shown in Figure 5). When equality of interference degrees is identified, a random node is chosen (it is the case for leaf nodes 5 and 6 or for leaf nodes 2 and 7).

node	1	2	3	4	5	6	7	8
interference degree	5	5	5	7	7	7	5	7

Fig. 5. Interference degree for nodes of the graph in Figure 4

The schedule in Figure 6 is obtained based on the IDeg-LO algorithm.



nodes	slots					
1			3	1	4	
2	5		2	7		8
3		6	3	1	4	
4	5		3	1	**4**	8
5	5	6	2	7	4	8
6	5	6	3			8
7	5		2	7		8
8	5	6	2	7	4	**8**

Fig. 6. TDMA scheduling for the graph in Figure 4 using the IDeg-LO algorithm

The order of the nodes to be scheduled is not always preserved in the schedule, because of two reasons:

- slots may be reused if no interference is generated (for example, for the schedule in Figure 6 node 3 uses the same slot as node 2),
- slot of predecessors are as close as possible to the last slot of its children, possibly cycling when no compatible slot exists after the children's slots (for example, for the schedule in Figure 6 node 1 uses the closest slot to the last slot of its children, that is node 4, by cycling because no slot is free after the slot of node 4).

VI. Interference degree remaining leaves order - IDeg-ReLO

The previous approach may schedule parent nodes before some of its successors. We observe in Figure 6 that node 8 is scheduled after node 4, which is its parent (their slots are bold in the figure). This makes node 4 wait for 5 slots before being able to further transmit data coming from node 8. Moreover, we also see that nodes with a great interference degree do not have the chance to be scheduled soon, because internal nodes are considered in the order of the successors in the tree. It is the case, in the same allocation example (see Figure 4) of node 8 whose interference degree is 7, while the interference degree of node 3 is 5, but this node is scheduled sooner.

We propose a second slot allocation heuristic where the order of the allocation is also based on the interference degree of nodes but the leaf considered to be scheduled is removed from the routing tree and the process of leaves allocation is repeated. This gives opportunity to internal nodes with great interference degree to be scheduled sooner. Moreover, as a result, nodes are never scheduled before their predecessors in the routing tree.

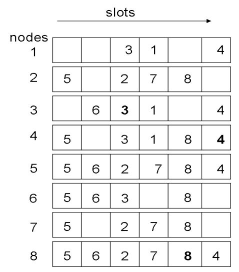
The scheduling algorithm, IDeg-ReLO, is summed up in the followings.

```
Schedule IDeg-ReLO(Graph<Node> graph,
            RoutingTree<Node> routingTree) {

 Schedule sched = new Schedule(
              graph.getMaxDegree()+1);
 RoutingTree<Node> tree = routingTree;

 do {
   ListOfNodes<Node> leaves =
      new ListOfNodes<Node>(tree.getLeaves());
   leaves.sortByInterferenceDegree(graph);
   Node n = leaves.removeFirst();
   sched.findFreeSlot(n, graph, routingTree);
   tree.remove(n);
 }while (!tree.isEmpty());
 return sched;
}
```

For the previous example, the order of the nodes considered to be scheduled according to the IDeg-ReLO scheduling algorithm, is the following: 5, 6, 2, 7, 8, 4, 3, 1 and the slot schedule is given in Figure 7.



nodes	slots					
1			3	1		4
2	5		2	7	8	
3		6	**3**	1		4
4	5		3	1	8	**4**
5	5	6	2	7	8	4
6	5	6	3		8	
7	5		2	7	8	
8	5	6	2	7	**8**	4

Fig. 7. TDMA scheduling for the graph in Figure 4 using the IDeg-ReLO algorithm

This still does not ensures that the allocation slot for a node will always preceed the allocation slot of its predecessor, even though, in the schedule obtained in Figure 7, slots of parents always come after the slots of their children.

VII. Scenario description and evaluated metrics

In this section, we present scenarios used for simulations. We used a Java-based library called JUNG [9] to model sensor networks and simulate routing algorithms. We generate random networks using $n = 100$ nodes with a communication range $r = 25$m deployed on a square area of size a^2. The sink is placed in one of the corners of the deployment area. The results represent averages for several simulation measures using different randomly generated networks.

One particular parameter of the network architecture was analyzed: its density. Based on the UDG model, the density of the generated networks is $\delta = \pi \times r^2 \times n/a^2$. We vary the a parameter to generate different network densities. For small densities (less than 8), the graph generator based on the UDG model generates 99% disconnected networks. In these cases, we used the graph generation proposed in [10].

Another parameter is induced by the routing tree used by the heuristics: the form of the routing tree. Three routing trees are used for simulations:

- the routing tree used by CoLaNet [6], which is the MinDegree tree. Each node chooses the node with the fewest children nodes as its parent.
- the hop count tree used by the Gradient-based routing protocol (GBRP) [11]. Using the flooding technique, each node keeps the number of hops between itself and the sink. This is considered to be the hop count metric. The difference between a node's hop count and the one of its neighbor is considered as the gradient of the link. A packet is forwarded on a link with the largest gradient.
- the geographic routing tree used by the greedy perimeter stateless routing (GPSR) [12], a common form of greedy forwarding in ad hoc networks. Packets contain the position of the destination and nodes need only local information about their position and their immediate neighbors' positions to forward the packets. Each wireless node forwards the packet to the closest neighbor to the destination among its neighbors (within radio range). We do not treat voids in this routing protocol.

The metrics used to compare approaches are the following.

- The average of the total delivery time (*average latency*) is computed as the average of all end-to-end delivery times (between each node, excluded the sink, and the sink, denoted by s); the delay is estimated in number of hops. Formally, let us consider the communication TDMA schedule of a node i: $sched_i$ of length l and its routing path, of length k_i (sink excluded), given by the ordered forwarding nodes $\{p_i^{j_1}, p_i^{j_2}, \cdots, p_i^{j_{k_i}}\}$. The corresponding routing path is: $i \rightarrow p_i^{j_1} \rightarrow p_i^{j_2} \rightarrow \cdots \rightarrow p_i^{j_{k_i}} \rightarrow s$. We consider latency for packets transmitted by every node to the sink. Concurrent flows are generated at the same time, at the first slot. Therefore, the delivery time for a packet generated at node i (considering that node i is not the sink), equals:
  $dt_i = s_{j_1} + (s_{j_2} - s_{j_1}) \bmod l + (s_{j_3} - s_{j_2}) \bmod l + \cdots + (s_{j_{k_i}} - s_{j_{k_i - 1}}) \bmod l$
  where $sched_i^{s_{j_v}} = j_v, \forall v \in \{1, 2, \cdots, k_i\}$.

  We specify that :

  $$(x-y) \bmod l = \begin{cases} (x-y) \bmod l, & \text{if } x \geqslant y \\ (x-y+l) \bmod l, & \text{if } x < y \end{cases}$$

  The average latency for paquets in a network of size n is defined as:

  $$dt = (\sum_{i=2}^{n} dt_i)/(n-1)$$

  considering that node 1 is the sink.
- The average of normalized delivery times (*average normalized latency*) is computed as the average of total delivery times per communication link.
  Formally, this metric needs to normalize the latency per path: $dtn_i = dt_i/k_i, \forall i, 2 \leqslant i \leqslant n$ (n being the size of the network) and to compute their average:

  $$dtn = (\sum_{i=2}^{n} dtn_i)/(n-1)$$
- The schedule length (denoted by l previously) is the number of slots used to schedule communications.

VIII. Simulation Results

A. Performance compared to CoLaNet

We compare the two approaches, IDeg-LO and IDeg-ReLO, to CoLaNet slot schedule as well as to the random slot schedule. In order to be able to compare results, the same routing tree is considered, that of CoLaNet (the min-degree tree). For every metric, average on thousands of runs are computed.

a) Average latency: is compared for CoLaNet, Random, IDeg-LO and IDeg-ReLO scheduling algorithms (see Figure 8), estimated for different densities (varying from 4 to 20).

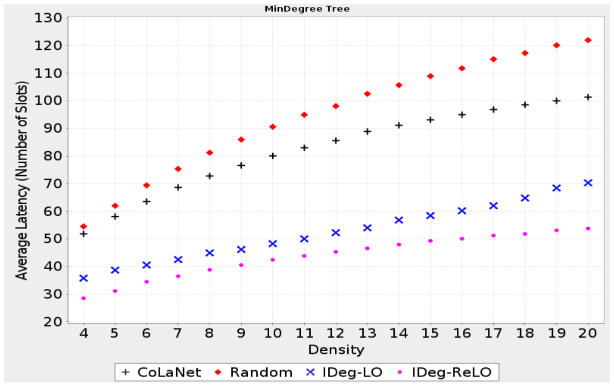


Fig. 8. Average latency based on the MinDegree routing tree for different scheduling approaches

IDeg-LO and IDeg-ReLO have the best performances for this metric, the former with an average gain of 44.14%, and the latter with an average gain of 53.33% compared to Random. CoLaNet is performing well, but the average gain is only 12.27% compared to Random (the average gain is the average of gains for every density value).

b) Average normalized latency: is shown in Figure 9. Gains are proportionaly the same, as the same routing tree is used.

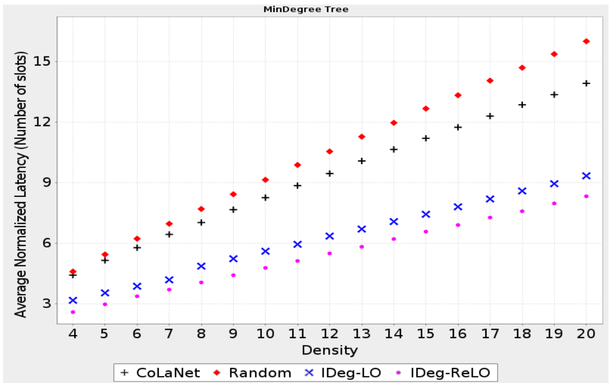


Fig. 9. Average normalized latency based on the MinDegree routing tree for different scheduling approaches

c) Schedule length: is minimal for CoLaNet (see Figure 10), with an average gain of 7.5% compared to the schedule length obtained by Random. IDeg-LO and IDeg-ReLO perform similarly in respect to this metric, average schedule length being longer of 4.5%, on average, than the Random schedule length. When compared to the CoLaNet schedule length, IDeg-LO schedules are 13.29% longer, on average, and IDeg-ReLO schedules are 12.94% longer on average.

IDeg-LO and IDeg-ReLO scheduling heuristics do not aim to minimize schedule length. The main objective is to obtain good latency, which is the case, even though the schedule length is longer than that of CoLaNet.

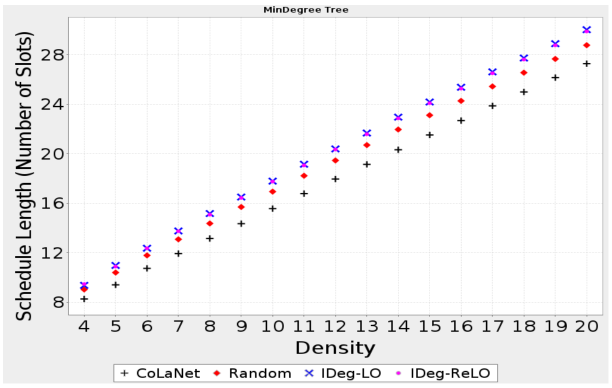


Fig. 10. Average schedule length for different scheduling approaches when using the MinDegree routing tree

B. Impact of the routing tree

We analyze the influence of the routing tree considered for our approaches. In order to be able to interpret results, the same randomly generated graphs together with their routing trees are used for all the scheduling algorithms. The routing tree has no influence for the random scheduling.

Figures 11 and 12 show the average total latencies when using the hop count routing tree, respectively the geographic routing tree. We notice that IDeg-ReLO gives, as previously, the best performances (having a gain of 33.53% compared to Random), but IDeg-LO is less efficient (with a gain of 13.19% compared to Random) and becomes close to CoLaNet (with a gain of 4.64% compared to Random). The previous observations apply also on the average normalized latency (see Figures 13 and 14).

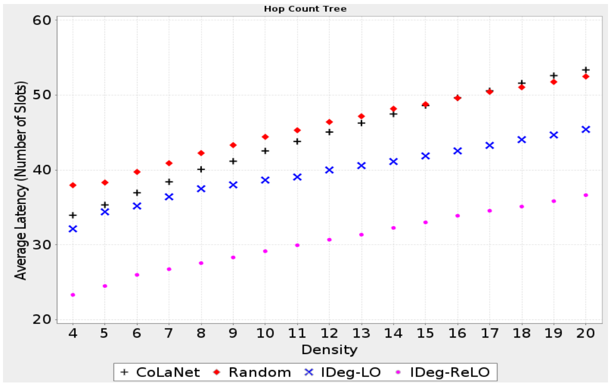


Fig. 11. Average latency based on the hop count routing tree for different scheduling approaches

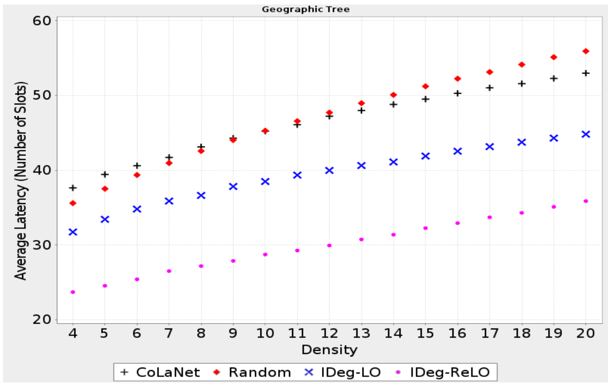


Fig. 12. Average latency based on the geographic routing tree for different scheduling approaches

CoLaNet and Random schedules give similar results in terms of average and normalized latencies, since both of them do not consider the latency metric when allocating the slots.

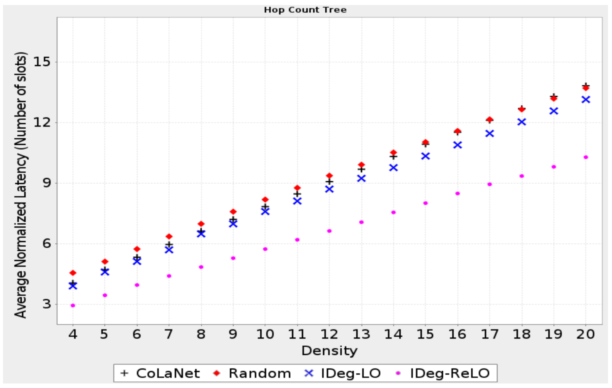


Fig. 13. Average normalized latency based on the hop count routing tree for different scheduling approaches

The average gains, irrespective to density, compared to Random normalized latencies are summed up in table I.

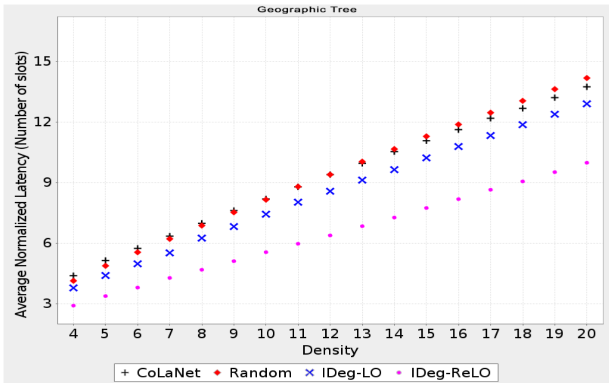


Fig. 14. Average normalized latency based on the geographic routing tree for different scheduling approaches

Average gains of normalized latency compared to Random	CoLaNet	IDeg-LO	IDeg-ReLO
MinDegree tree	9.91%	39.30%	47.45%
hop count tree	3.37%	7.52%	29.06%
geographic tree	2.17%	9.3%	31.21%

TABLE I. Gains of normalized latency obtained by CoLaNet, IDeg-LO and IDeg-ReLO compared to Random depending on the routing tree

The schedule length is close to that obtained by CoLaNet (see Figures 15 and 16), which is the best of the four heuristics. This was expected because CoLaNet is using an approximation algorithm for the vertex coloring problem, which tends to reduce the number of slots used for the schedule. The two routing-based heuristics perform well when using the hop count routing tree (average losses of 7.2% for IDeg-LO and of 6.48% for IDeg-ReLO). Average losses are similar when using the geographic routing tree: 8.39% for the IDeg-LO slot scheduling and 6.63% for the IDeg-ReLO slot scheduling.

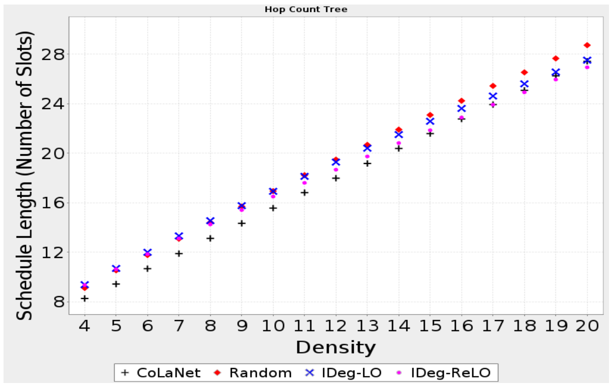


Fig. 15. Average schedule length for different scheduling approaches when using the hop count routing tree

IX. Conclusions and Future Work

This paper presents IDeg-LO and IDeg-ReLO, new TDMA scheduling techniques for sensor nodes, using information of the routing tree and a particular node ordering based on the interference degree of each sensor node.

The slots are allocated to each node using information from the routing tree to gain latency. The interference degree is used to first schedule nodes with high interference. Consequently, latency for every node communication is reduced.

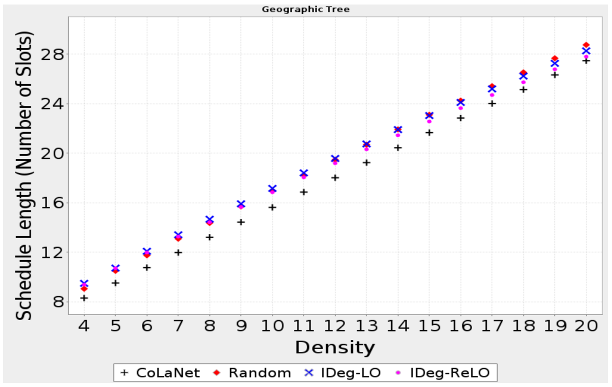


Fig. 16. Average schedule length for different scheduling approaches when using the geographic routing tree

Compared to the state of the art, results show that IDeg-LO improves average latency up to 44.14%, while it is improved up to 53.33% with IDeg-ReLO which demonstrates the importance of the routing information and the neighborhood of a node when designing its schedule.

A number of open problems arise from this work; we may analyze the behavior of the two methods if another metric than the interference degree were used for node ordering.

References

[1] L. D. Mendes and J. J.P.C. Rodrigues, "A survey on cross-layer solutions for wireless sensor networks," *Journal of Network and Computer Applications*, vol. 34, no. 2, pp. 523–534, Mar. 2011.

[2] T. Melodia, M. C. Vuran, and D. Pompili, "The state of the art in cross-layer design for wireless sensor networks," in *Wireless Systems and Network Architectures in Next Generation Internet*. Springer, 2006, pp. 78–92.

[3] C. Suh, Y.-B. Ko, and D.-M. Son, "An energy efficient cross-layer mac protocol for wireless sensor networks," in *APWeb Workshops*, 2006, pp. 410–419.

[4] M. Hefeida, T. Canli, and A. A. Khokhar, "CL-MAC: A cross-layer mac protocol for heterogeneous wireless sensor networks," *Journal of Ad Hoc Networks*, vol. 11, no. 1, pp. 213–225, 2013.

[5] K. Heurtefeux, F. Maraninchi, and F. Valois, "Areacast: A cross-layer approach for a communication by area in wireless sensor networks," in *International IEEE Conference on Networks (ICON)*, 2011, pp. 112–117.

[6] C.-F. Chou and K.-T. Chuang, "CoLaNet: A Cross-Layer Design of Energy-Efficient Wireless Sensor Networks," *Wireless Technologies/High Speed Networks/Multimedia Communications Systems/Sensor Networks, International Conference on*, pp. 364–369, 2005.

[7] S. Schmid and R. Wattenhofer, "Algorithmic models for sensor networks," in *14th WPDRTS*, 2006, pp. 450–459.

[8] S. Pemmaraju and S. Skiena, *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*. NY, USA: Cambridge University Press, 2003.

[9] J. O'Madadhaina, D. Fisher, P. Smyth, S. White, and Y.-B. Boey, "Analysis and visualization of network data using JUNG," *Journal of Statistical Software*, vol. 10, no. 2, pp. 1–35, 2005.

[10] F. A. Onat and I. Stojmenovic, "Generating Random Graphs for Wireless Acuator Networks," in *IEEE WoWMoM*, 2007, pp. 1–12.

[11] C. Schurgers and M. Srivastava, "Energy efficient routing in wireless sensor networks," in *IEEE Military Communications Conference (MILCOM)*, 2001, pp. 357–361.

[12] B. Karp and H. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *6th Intern. Conf. on Mobile computing and networking*, 2000, pp. 243–254.