OSERENA: a Coloring Algorithm Optimized for Dense Wireless Networks

Ichrak Amdouni, Pascale Minet, Cédric Adjih

OSERENA: a Coloring Algorithm Optimized for Dense Wireless Networks

Ichrak Amdouni, Pascale Minet, Cedric Adjih

INRIA,
Rocquencourt, 78153 Le Chesnay cedex, France
Email: ichrak.amdouni@inria.fr, pascale.minet@inria.fr, cedric.adjih@inria.fr

Abstract

The goal of this paper is to present OSERENA, a distributed coloring algorithm optimized for dense wireless sensor networks (WSNs). Network density has an extremely reduced impact on the size of the messages exchanged to color the WSN. Furthermore, the number of colors used to color the network is not impacted by this optimization. We describe in this paper the properties of the algorithm and prove its correctness and termination. Simulation results point out the considerable gains in bandwidth.

Keywords: Wireless networks, ad hoc networks, wireless sensor networks, node coloring, distributed coloring, dense networks, bandwidth efficiency.

1. Introduction and state of the art

Graph coloring can be seen as a specific case of graph labeling: labels, usually called colors, are assigned to vertices (respectively edges) of a graph subject to certain constraints. Depending on which graph element is colored, we obtain vertex or edge coloring. For both, the objective is to minimize the number of colors needed to color the whole graph. Typically, the constraint considered for h-hop vertex coloring, with h an integer $\geqslant 1$ is: no two vertices that are k-hop neighbors with $1 \leqslant k \leqslant h$ have the same color. The Vizing's theorem [1] states that the minimum number of colors needed to 1-hop color a graph, number denoted χ, meets $\Delta \leqslant \chi \leqslant \Delta + 1$, where Δ is the maximum degree of the graph. The h-hop node coloring problem has been proved NP-complete in [2] for $h = 1$ and in [3, 4] for any $h > 1$. This explains why heuristics are used for large graphs. Authors of [5] compare the performances of different heuristics for edge coloring over standard benchmarks (taken from a list of 119 graphs given at CP2002) for small graphs (< 500 nodes). They show that among the tested heuristics, *ant*1 often finds the optimum or a number of colors differing not much. A survey of local search methods (*TabuCOL*, simulated annealing, neighborhood search, and clustering guided search) can be found in [6] and [7]. If these algorithms are efficient for small graphs, it is no longer the case with large random graphs [7]. Hybrid algorithms can be used, as well as the extraction of large independent sets from the graph to obtain a smaller residual graph easier to color (see for instance *EXTRACOL* that needs 2.5 hours to color 1000 nodes with a density of 5 in [7]). The main performance criteria of a coloring algorithm are the number of colors and the time needed to color the considered graph. Of crucial interest is the approximation ratio of coloring algorithms that is defined as the ratio of the number of colors obtained by the algorithm to the optimal number. A well-known coloring algorithm is FirstFit [8] that sequentially assigns colors to nodes. Each node is colored with the first available color. Depending on the coloring order, different coloring results are obtained. Approximation ratio of coloring algorithms for grids, triangular lattices and hexagonal graphs can be found in [10]. All nodes having the same color constitute a class. Hence, graph coloring can also be seen as determining independent sets of maximum size.

Coloring has been applied to wireless networks to improve medium access efficiency. Thus, with node coloring, nodes access the medium in time slots corresponding to their color [11–13]. Only nodes that do not interfere can transmit simultaneously, hence collisions are avoided while spatial reuse of the bandwidth is provided. The smaller the number of colors, the shorter the activity period in data gathering applications [14–16]. A color can be mapped into a channel, that is why graph coloring has been applied to channel assignment reducing radio interferences [17].

Running a distributed coloring algorithm on WSNs (Wireless Sensor Networks) is very challenging because of their strong limitations. They have low capacity of storage and computing, low energy especially for battery operated nodes and the network bandwidth is also limited. That is why algorithms supported by WSNs must be of low complexity. More challenging are dense WSNs, where a node cannot maintain its 2-hop neighbors because of memory limitation and a single message cannot contain all the information relative to the 2-hop neighbors of a node. Examples of dense WSNs are given by smart dust where microelectomechanical systems called MEMS can measure temperature, vibration or luminosity. Applications can be monitoring of building temperature, detection of seismic events, monitoring of pollution, weather prediction for vineyard protection... In this paper, we show how to optimize a coloring algorithm for dense WSNs. We present OSERENA, an optimized version of the node coloring algorithm SERENA [18]. The optimization consists in the reduction of the algorithm overhead in both sizes of data stored and messages exchanged to color the network. Indeed, OSERENA does not require neither the storage nor the exchange of neighbors up to two hops. Furthermore, we prove that OSERENA keeps the same number of colors as SERENA. Moreover, OSERENA produces a small convergence time that is equal, most of the time, to the time needed by SERENA to color the algorithm.

The paper is organized as follows. In Section 2, we present OSERENA, a 3-hop node coloring algorithm that is optimized for dense networks. In section 3, we present the properties of OSERENA regarding the correctness, the overhead induced, and its convergence time. We also prove that OSERENA is equivalent to a centralized version of 3-hop node coloring. In Section 4, we evaluate the performance of OSERENA for many network configurations, varying network size and node density. We show that unlike the previous work in [18], OSERENA keeps a number of rounds similar to the number of rounds induced by SERENA (the unoptimized version) to color the network, while using smaller messages. This property is illustrated through an extensive performance evaluation by means of simulations. Finally, we conclude in Section 5.

2. Coloring optimized for dense networks

The goal of this section is to make possible the use of the coloring algorithm in dense wireless sensor networks. We show how to reduce the overhead in terms of 1) memory required to store the data maintained by each node and 2) bandwith used by exchanging messages between neighbors. Of course this overhead reduction must not decrease the performance of the coloring algorithm: the number of colors and the time needed to color all network nodes must be kept small. First, we give some definitions followed by the basic principles of 3-hop node coloring.

2.1. Assumptions and definitions

2.1.1. Central assumptions and definitions

The type of node coloring needed to support a given application depends on the type of:

- *communications supported*: unicast and/or broadcast;

- *application*: general where any node is likely to exchange information with any neighbor node or on the contrary tree type where a node exchanges information only with its parent and its children in the data gathering tree;
- *acknowledgement for unicast transmissions*: immediate or deferred.

In this paper, we focus on 3-hop node coloring, which was proved in [18] to be necessary to support general communications, where unicast transmissions are immediately acknowledged. We assume an ideal environment where:
Assumption A0: All links are symmetric and stable.
Assumption A1: Each node has a unique address in the network.
Assumption A2: Any node does not prevent the correct receipt of any other node out of its transmission range.

A 3-hop node coloring is said *valid* if and only if no two 1-hop, 2-hop or 3-hop nodes have the same color. The smaller the number of colors obtained, the better the coloring algorithm.

The time complexity of a coloring algorithm is generally evaluated in terms of rounds. By definition, a *round* is such that any node receives the messages sent by its 1-hop neighbors, processes them and broadcasts its own message to its 1-hop neighbors. The space complexity is given by the number and size of messages sent per node.

2.1.2. Further assumptions and simplifications

In the some sections, we will assume a more specific model, closely related to a common model for wireless sensor networks: the unit disk graph model [9]. Hence:

- Nodes are modeled as a set of points in the 2-dimensional plane.
- A uniform transmission range R is defined.
- A node receives a transmission from another node, if and only if, its distance is lower than R.
- There are no losses.

The same model is applied for instance for simulations in section 4.

Furthermore, in some calculation, we also make the following approximation:
Assumption (approximation) A3: we equate distance to number of hops (e.g. a node at distance between R and $2R$ from another node, is assumed to be at 2 hops).
The assumption is valid asymptotically when the density converge towards infinity ; for a more detailed exploration of the exact relationship between number of hops and distance, see for instance [19].

2.2. Basic principles of 3-hop node coloring

In SERENA, any node u proceeds as follows to color itself:

1. Node u characterizes the set $\mathcal{N}(u)$ of nodes that cannot have the same color as itself. The set $\mathcal{N}(u)$ is the set of neighbors up to 3-hop from u in 3-hop node coloring.

2. Node u computes its priority, denoted $priority(u)$. This priority consists of two components: the most important one is denoted $prio(u)$. It can be equal to the number of nodes up to 2-hop (resp. 3-hop) from u. We will see later the exact value taken in OSERENA. The second component of $priority(u)$ denotes the *address* of the node. By definition, node u is said to have a priority higher than node v if and only if:

   - either $prio(u) > prio(v)$;
   - or $prio(u) = prio(v)$
     and $address(u) < address(v)$.

3. Node u applies the two following rules:

   - **Rule R1**: Node u colors itself if and only if it has a priority strictly higher than any uncolored node in $\mathcal{N}(u)$.
   - **Rule R2**: To color itself, node u takes the smallest color unused in $\mathcal{N}(u)$.

2.3. Motivations and optimization principles

This distributed coloring algorithm proceeds by iterations or rounds, where nodes exchange their *Color* message. In its simplest implementation, the *Color* message would include the address, the priority and the color of 1) the node u itself, 2) its 1-hop neighbors in $\mathcal{N}(u)$, as well as 3) its 2-hop neighbors in $\mathcal{N}(u)$. The data locally maintained by any wireless sensor would include these data as well as the priority and color of any neighbor up to 3-hop. It is well known that the average number of nodes in the neighborhood up to 2-hop is equal to $4 \cdot density$, where $density$ stands for the average number of nodes in the disk of radius R, where R is the transmission range. Such an overhead can be unacceptable for wireless sensors with limited storage and processing capabilities as well as low residual energy. Dense networks with limited bandwidth, low energy and a short MAC frame size become challenging for a coloring algorithm. That is why, we propose in this paper an optimization of the coloring algorithm reducing the size of *Color* messages exchanged and the size of data structures maintained, while keeping a low complexity. We also show that this overhead reduction does not increase the convergence time of the coloring algorithm. The optimization principles are based on the following remarks:

- It is necessary that any node u knows the highest priority taken by its uncolored neighbors up to 3-hop in order to apply Rule R1. Furthermore, node u must send information concerning itself, its 1-hop and 2-hop neighbors to let its one-hop neighbors know information about their 1-hop, 2-hop and 3-hop neighbors. Hence, node u must send its priority, the highest priority taken by its uncolored 1-hop neighbors as well as the highest priority taken by its uncolored 2-hop neighbors. However, sending only one highest priority of the uncolored 1-hop or 2-hop neighbors would delay the coloring since the information update will be slow. This would not suffice to color any wireless network with the same number of rounds as SERENA. Indeed, node v, 2-hop away from node u colored at round r would not know at round $r+2$ that it has the highest priority. Hence, MORE THAN ONE highest priority at respectively 1-hop and 2-hop must be maintained and sent, unlike the version briefly presented in [18]. We will wee in Section 3.3 how to compute the near optimal number of priorities to maintain at one-hop and two-hop respectively. Notice that the highest priority at 3-hop is locally computed and not sent.
- Similarly for the color, node u must know the colors already used in its neighborhood up to 3-hop. However, it does not matter u to know which node up to 3-hop has which color, but only which colors are taken at 1-hop, 2-hop and 3-hop respectively. That is why, we use the fields $color\_bitmap1$, $color\_bitmap2$ and $color\_bitmap3$ for the bitmaps of colors used at 1-hop, 2-hop and 3-hop respectively.

2.4. OSERENA: Optimized coloring algorithm

2.4.1. The Color message

From simulation feedback, we have noticed that the assignment of $prio(u)$ =number of neighbors up to 2 hops outperforms the assignment $prio(u)$ =number of neighbors up to 3 hops from u, or a random assignment. However, as OSERENA avoids the expensive computation of the list of neighbors up to 2 hops, OSERENA defines for any node u, $prio(u)$ as the number of its neighbors + the sum of the number of 1-hop neighbors of its 1-hop neighbors. This computation is done during the initialization of the coloring algorithm. We also define $max\_prio1(u)$ as:

- the four highest priorities of the uncolored 1-hop neighbors of u, if four such nodes exist;
- the priority of the only three (respectively two, respectively one) uncolored 1-hop neighbor, if only three (respectively two, respectively one) such nodes exist;
- empty, denoted $\varnothing$, if none exists.

We then have the following notation:
$max\_prio1(u) = Max4_{\ v\ uncolored \in 1hop(u)}\ priority(v)$.

Similarly, we define $max\_prio2(u)$ as the three highest priorities of the uncolored 1-hop neighbors of the

1-hop neighbors of u, if they exist. We then have: $max\_prio2(u) = Max3_{v \in 1hop(u)}\ max\_prio1(v)$.

The variable $max\_prio3(u)$ is defined as the highest priority of the uncolored 1-hop neighbors of the 1-hop neighbors of the 1-hop neighbors of u. We get: $max\_prio3(u) = Max_{v \in 1hop(u)}\ max\_prio2(v)$.

The computation of $max\_prio1(u)$, $max\_prio2(u)$ and $max\_prio3(u)$ is done from the *Color* messages received during the current round. The values computed for $max\_prio1(u)$ and $max\_prio2(u)$ are inserted in the *Color* message sent by node u.

It follows that the *Color* message sent by any node u contains $priority(u)$, $max\_prio1(u)$ and $max\_prio2(u)$, as well as the color of u, the bitmap of colors used at 1-hop from u, denoted $bitmap1(u)$ and the bitmap of colors used at 2-hop from u, denoted $bitmap2(u)$.

2.4.2. *Processing*

With the optimization, Rules R1 and R2 become:

Rule R'1: Any node u colors itself if and only if:
$priority(u) = max(max\_prio1(u), max\_prio2(u), max\_prio3(u)).\ (eq.1)$

Rule R'2: When a node u selects its color, it selects the smallest color unused in $color\_bitmap1(u) \cup color\_bitmap2(u) \cup color\_bitmap3(u)$.

Notice that this color should also not be used by heard nodes (nodes with which there is no symmetric link). This, in order to avoid color conflicts.
The aim of rules R3 and R4 is to improve convergence time. Although the *Color* message does not contain the whole list of colored 2-hop neighbors, a node processing this message can deduce the recently colored nodes and stores them in a local data structure denoted $implicit\_node\_colored\_list$ whose size is equal to $implicit\_node\_colored\_size$. As we will see later, the storage of this list decreases the coloring time. To build this list, any node u proceeds as follows:

Rule R3: When a node u receives the *Color* message from any neighbor node v it compares the current value of $max\_prio1(v)$ (respectively $max\_prio2(v)$) with the previous one sent by v, denoted $previous\_max\_prio1(v)$ (respectively $previous\_max\_prio2(v)$). Any priority value of $previous\_max\_prio1(v)$ (respectively $previous\_max\_prio2(v)$) higher than the highest value of $max\_prio1(v)$ (respectively $max\_prio2(v)$) corresponds to a recently colored node. This node is then inserted in the set $implict\_node\_colored\_list$.

Rule R4: When a node computes max_prio1, max_prio2 and max_prio3 from the values received in the *Color* messages, it proceeds as follows:

- in the computation of max_prio1, it discards any priority value corresponding to an already colored node (that is a node that belongs to the list $implicit\_node\_colored\_list$).
- in the computation of max_prio2, it discards for any sender v, any priority value p corresponding to an already colored node received in $max\_prio1(v)$ if and only if:
  1. either p is the highest priority in $max\_prio1(v)$,
  2. or p is the second highest priority in $max\_prio1(v)$ and (the third or fourth highest priority in $max\_prio1(v)$ is equal to $\varnothing$),
  3. or p is the third highest priority in $max\_prio1(v)$ and (the fourth highest priority in $max\_prio1(v)$ is equal to $\varnothing$).
- in the computation of max_prio3, it discards for any neighbor v, any priority value corresponding to an already colored node received in $max\_prio2(v)$ if and only if it is the highest or the second highest priority in $max\_prio2(v)$.

The motivation of this rule is that a node u can receive information about a node v from a neighbor w such that the distance between v and w is greater than the distance between v and u. Consequently, node w can send node v as an uncolored node while

u knows that this node is colored. In such a case, if u considers this information from v and it sends it to its neighbors, coloring will be delayed because the node to be colored after w will think this latter is uncolored and so does not color itself. However, using the list *implicit_node_colored_list*, u will discard the node w and does not propagate an out-of-date information which helps to speed up the coloring convergence time. However, not any colored node can be discarded from max_prio1 or max_prio2. Indeed, let us consider a node u that discards the 4 values sent in $max\_prio1(v)$. In some configurations, u might think it is the node having the highest priority among its 3 hop neighbors, although v would have sent a priority higher than $priority(u)$ if it could send more than 4 values in a max_prio1 it sends. That is why, we adopted the Rule 4. A node can be discarded from max_prio1 or max_prio2 if there is still at least one priority that may be equal to $\varnothing$.

Rule R5 is related to the termination rule of the coloring algorithm.

Rule R5: Any node u stops sending its *Color* message as soon as it is colored, $max\_prio1(u) = \varnothing$ and it has received from all its 1-hop neighbors v a *Color* message with $max\_prio1(v) = max\_prio2(v) = \varnothing$.

Rule R6 has been introduced to tolerate message losses and link failures.

Rule R6: If at a round $r > 1$ of the coloring algorithm, any node u does not receive a message from its 1-hop neighbor v, it uses the information received from v at round $r-1$. After n successive rounds, with $n \geqslant 2$ without receiving a *Color* message from v, v is no longer considered as a 1-hop neighbor of node u.

3. Properties of OSERENA

3.1. Correctness of OSERENA coloring

In this section we prove that in a wireless environment assuming hypothesis A0, A1 and A2, OSERENA provides a valid 3-hop node coloring avoiding collisions. Furthermore, we prove that this algorithm ends when all nodes are colored.

Lemma 1. *With OSERENA, any node u colors itself if and only if it has the highest priority among all the uncolored nodes in $\mathcal{N}(u)$.*

Proof. Let us show that if any node u is coloring itself, then it has the highest priority among the uncolored nodes up to 3-hop. By Rule R'1, if u is coloring itself then $priority(u) = max(max\_prio1(u), max\_prio2(u), max\_prio3(u))$ $(eq.1)$.
From $(eq.1)$, we get $priority(u) \geqslant max(max\_prio1(u))$. Hence, no uncolored one-hop neighbor has a priority higher than u.
From $(eq.1)$, we get $priority(u) \geqslant max(max\_prio2(u))$. Hence, no uncolored two-hop neighbor has a priority higher than u, otherwise we would have the following contradiction:
$priority(u) \geqslant max(max\_prio2(u)) > priority(u)$.
From $(eq.1)$, we get $priority(u) \geqslant max\_prio3(u)$. Hence, no uncolored three-hop neighbor in $\mathcal{N}(u)$ has a priority higher than u, otherwise we would have the following contradiction: $priority(u) \geqslant max\_prio3(u) > priority(u)$.
Hence, node u has the highest priority among the uncolored nodes in $\mathcal{N}(u)$.

Conversely, if node u has the highest priority among its uncolored neighbors up to 3-hop, it means that:

- all its uncolored one-hop neighbors have a smaller priority. Hence, for any v uncolored one-hop neighbor of u, we have $priority(v) < priority(u)$. Hence, $max(max\_prio1(u)) = max_{v \in 1hop(u)}(priority(v) \;\; for \;\; v \;\; uncolored \;\;) < priority(u)$;
- all its uncolored two-hop neighbors have a smaller priority. Let us consider the highest priority in $max\_prio2(u)$. It denotes the highest priority of an uncolored node w that is one-hop neighbor of v, itself one-hop neighbor of u. Consequently, we have the following cases:
  - node w is the node u itself and has priority $priority(u)$;

  - node w is a one-hop or two-hop neighbor of node u. In which case, we have by assumption: $priority(w) < priority(u)$.

  Hence, $max(max\_prio2(u)) = priority(u)$.

- and all its uncolored three-hop neighbors in $\mathcal{N}(u)$ have a smaller priority. By definition, $max\_prio3(u)$ is the maximum priority of uncolored nodes q that are one-hop neighbors of w, itself one-hop neighbor of v, one-hop neighbor of u. Consequently, we have the following cases:
  - node q is the node u itself and has priority $priority(u)$;
  - node q is a one-hop, two-hop or three-hop neighbor of node u. In which case, we have by assumption: $priority(q) < priority(u)$.

  Hence, $max\_prio3(u) = priority(u)$.

Finally, $priority(u) = max(max\_prio1(u), max\_prio2(u), max\_prio3(u))$.
Hence, node u is coloring itself with OSERENA. □

Lemma 2. *With OSERENA, when node u colors itself, it knows all the colors taken in $\mathcal{N}(u)$ with a higher priority.*
Proof. The exchange of *Color* messages allows any node u to know any uncolored node in $\mathcal{N}(u)$ having a higher priority than itself. Node u also knows the colors of already colored nodes in $\mathcal{N}(u)$ by means of $bitmap1$, $bitmap2$ and $bitmap3$. Thus, when u colors itself, it takes the smallest color unused in these bitmaps, and hence unused in $\mathcal{N}(u)$. □

Lemma 3. *OSERENA coloring ends when all nodes are colored.*
Proof. If u is colored and $max\_prio1(u) = \varnothing$, then node u and all its one-hop neighbors are colored. Moreover, if node u receives a *Color* message from any one-hop neighbor v with $max\_prio1(v) = max\_prio2(v) = \varnothing$, it means that all the one-hop neighbors of v and all the one-hop neighbors of its one-hop neighbors are already colored. Hence, all nodes up to three-hop from u and belonging to $\mathcal{N}(u)$ are colored. The coloring algorithm ends when node u as well as all its 1-hop, 2-hop and 3-hop neighbors are colored. □

Lemma 4. *In a wireless network meeting assumptions A0, A1 and A2 and in the absence of message loss and node failure, all nodes color themselves with OSERENA and stop sending their Color message.*
Proof. Let us consider any node u. The nodes in $\mathcal{N}(u)$ color themselves according to their priority. As soon as u becomes the uncolored node with the highest priority, it colors itself according to rules R'1 and R'2. According to rule R5, as soon as u is colored and $max\_prio1(u) = \varnothing$, then node u and all its one-hop neighbors are colored. Moreover, if node u receives a *Color* message from any one-hop neighbor v with $max\_prio1(v) = max\_prio2(v) = \varnothing$, it means that all the one-hop neighbors of v and all the one-hop neighbors of its one-hop neighbors are already colored. Hence, all nodes up to three-hop from u and belonging to $\mathcal{N}(u)$ are colored. Hence, it is useless for u to send its *Color* message insofar as any information contained in its message is already known by its one-hop, two-hop and three-hop neighbors in $\mathcal{N}(u)$ and these nodes are already colored. □

Property 1 *OSERENA provides a valid 3-hop node coloring in any ideal wireless environment.*

Proof. For three-hop coloring, for any node u, the set $\mathcal{N}(u)$ contains by definition all nodes up to 3-hop from u, assuming an ideal environment. From Lemma 1, with three-hop coloring, any node u can color itself if and only if no uncolored node in $\mathcal{N}(u)$ has a priority higher than u.
According to rule R'1, priority of node u meets $(eq.1)$. Moreover, since no two nodes have the same priority, we cannot have a simultaneous coloring of two nodes up to 3-hop away each other. According to Lemma 2, when coloring itself, any node u knows all the colors taken by nodes in its $\mathcal{N}(u)$, so it selects the smallest color according to rule R'2. Consequently, assuming an ideal wireless environment, no 2 nodes within 3-hop neighborhood from

each other takes the same color. Which means that OSERENA provides a valid coloring. With this coloring, nodes that belong to $\mathcal{N}(u)$ cannot create a collision with data sent by u or an acknowledgement sent to u. □

Property 2 *A failure to receive a Color message from a one-hop neighbor induces an additional latency in network coloring and does not compromise the validity of coloring with OSERENA.*

Proof. Deduced from rule R6. □

3.2. Equivalence of OSERENA to a centralized algorithm

In this section, we compare the behavior of OSERENA with the well-known centralized First Fit 3-hop node coloring [8]. More precisely, we compare the colors granted to nodes by both coloring algorithms. With centralized First Fit 3-hop node coloring, nodes are sorted according to their priority and are colored in that order. Any node u receives the smallest unused color in $\mathcal{N}(u)$.

Lemma 5. *For any node u, for any given priority assignment, nodes $\in \mathcal{N}(u)$ color themselves in the same order with OSERENA and First Fit.*
Proof. Let us consider any node u that is coloring itself in OSERENA, we have:

- any node $v \in \mathcal{N}(u)$ such that $priority(v) > priority(u)$ is already colored in OSERENA, otherwise u could not color itself now;
- any node $v \in \mathcal{N}(u)$ such that $priority(v) < priority(u)$ is not colored in OSERENA, because it is constrained by node u that is not yet colored.

Hence, in $\mathcal{N}(u)$ the coloring order in OSERENA is compliant with the priority order that is by definition followed by First Fit. In conclusion, both coloring algorithms follow the priority order to color nodes in a given neighborhood $\mathcal{N}(u)$. □

Property 3 *For any topology, OSERENA provides the same coloring as a centralized First Fit 3-hop node coloring algorithm using the same priority assignment.*

Proof. For any topology, for any node u in this topology, the color of u is determined by the colors already used in $\mathcal{N}(u)$ when u colors itself. According to Lemma 5, all nodes in $\mathcal{N}(u)$ color themselves in the same order with OSERENA and First Fit. Let u_1 be the first node that colors itself in $\mathcal{N}(u)$. It takes the smallest available color in $\mathcal{N}(u_1)$. Let u_2 be the the first node that colors itself in $\mathcal{N}(u_1)$, and so on. After a finite number of iterations (at most equal to the number of nodes in the topology), we get a node u_{k+1} the first node that colors itself in $\mathcal{N}(u_k)$ and has colored itself without being constrained by any other node in OSERENA: u_{k+1} has the highest priority in $\mathcal{N}(u_{k+1})$. This node takes the color 0 in OSERENA. With First Fit, since no node in $\mathcal{N}(u_{k+1})$ is already colored, u_{k+1} takes color 0. Nodes in $\mathcal{N}(u_{k+1})$ with a priority higher than or equal to $priority(u_k)$ are colored according to their priority order with OSERENA and First Fit. Consequently, they receive the same colors. We apply the same reasoning to node u_k and nodes in $\mathcal{N}(u_k)$ with a priority higher than or equal to $priority(u_{k-1})$, going back up to node u_1 and finally node u that receives the same color with OSERENA and First Fit, because the same colors are already assigned in $\mathcal{N}(u)$. □

3.3. Reduced overhead

In this section, we show how OSERENA reduces the overhead both in terms of 1) bandwidth by reducing message number and message size and 2) node storage by decreasing the size of data maintained at each node.

OSERENA does not require to send or to maintain the 2-hop neighborhood of a node, as shown in Section 2.4.1. The use of max_prio1 and max_prio2 reduces the size of *Color* messages exchanged between neighbors. We now show how to determine the optimal size of max_prio1 and max_prio2.

3.3.1. Message size

Assuming the near optimal size of max_prio1 and max_prio2 determined later on (see Lemma 6), we can compute the maximum size of the message *Color* exchanged between neighbor nodes.

Property 4 *With the setting* $Size\_max\_prio1 = 4$ *and* $Size\_max\_prio2 = 3$*, OSERENA uses a Color message whose size is at most* $8 \cdot (size\_address + size\_prio) + size\_color + size\_bitmap1 + size\_bitmap2$ *bytes.*

Proof. This is deduced from the *Color* message format, where the 8 factor comes the maximum size of $priority + max\_prio1 + max\_prio2$. □

3.3.2. Constraints for the computation of max_prio1 and max_prio2 sizes

We first notice that the reduction of message size must not imply a higher number of rounds to color the network. Hence, the optimal size of max_prio1 and max_prio2 is a trade-off between bandwidth consumption and convergence time of the coloring algorithm.

The simplest solution would be to maintain only one priority for max_prio1 and max_prio2. However, this solution does not allow to remove already colored nodes in the computation of max_prio1, max_prio2 and max_prio3. Hence, a coloring that is much slower than SERENA. That is why, several priorities are maintained in max_prio1 and max_prio2. The question is how many? To be able to discard one value corresponding to an already colored node and sent by neighbor v in $max\_prio2(v)$ implies that v sends at least 2 values in $max\_prio2(v)$. To be able to compute its 2 highest values in $max\_prio2(v)$ and discard one value, node u must receive at least 3 values in $max\_prio1(u)$. Hence, the minimum sizes are $Size\_max\_prio1 = 3$ and $Size\_max\_prio2 = 2$.

Unfortunately, we can still exhibit scenarios with this minimum setting, where only the first address in max_prio2 is discarded if already colored, producing a number of rounds higher than SERENA. In simulations, we identified a scenario with 100 nodes uniformly distributed with a density of 20 need 175 rounds to color themselves with OSERENA instead of 134 rounds with SERENA. That is why, we select $Size\_max\_prio1 = 4$ and $Size\_max\_prio2 = 3$.

3.3.3. Computation of the optimal size of max_prio1 and max_prio2

In this section, we assume the unit disk graph model of section 2.1.2 (including *Assumption A3*).

Recall that in OSERENA, any node u receiving a *Color* message from one neighbor v, can have fresher information than v. That is, v believes that node w is not yet colored and so, keeps it in $max\_prio1(v)$ or $max\_prio2(v)$ it sends, whereas u knows that w is already colored because it it closer to w than v. In such a case, OSERENA allows u to ignore w when computing $max\_prio1(u)$ by usage of the list *implicit_node_colored_list* as explained in section 2.4.2. However, to keep the correctness of the algorithm, the node u cannot always ignore the colored node w when computing $max\_prio2(u)$ and $max\_prio3(u)$ (see the coloring rule R3). If node u that is the next node to be colored after w, is not allowed to ignore w already colored, u will not color itself and will wait until node v removes node w. Hence, node u in OSERENA colors itself later than it would do in SERENA.

Lemma 6. *With the setting* $Size\_max\_prio1 = 4$ *and* $Size\_max\_prio2 = 3$ *and rules R3 and R4, OSERENA colors any node u in the same round as SERENA, except when three nodes two-hop away from u, but 4-hop away from each other are coloring simultaneously just before u.*

Proof. We first identify this scenario and then compute its probability in the next section. When three nodes two-hop away from u, but 4-hop away from each other are coloring simultaneously just before u, node u is not allowed by rule R3 to discard the three of them in the received $max\_prio2(v)$, hence the coloring of node u is delayed. We can show that this scenario is the only one that will delay u coloring. On the one hand, two one-hop neighbors of u are not allowed to color simultaneously, because they are at most two-hop away. On the other hand, a one-hop and a two-hop neighbor of u are not allowed to color simultaneously, because they are at most three-hop away. It results that the only case of simultaneous colorings in $\mathscr{N}(u)$ involves nodes that are 2-hop away from u and 4-hop away from each other. $\square$

Lemma 7. *The setting $Size\_max\_prio1 = 5$ and $Size\_max\_prio2 = 4$ provides the same number of rounds as SERENA.*

Proof. With the setting, $Size\_max\_prio1 = 5$ and $Size\_max\_prio2 = 4$, it is no longer possible to have a bad scenario where four nodes two-hop away from u, but 4-hop away from each other are coloring simultaneously. We prove it by contradiction. Let u be any node. We assume that the four nodes v_1, v_2, v_3 and v_4 that are 4-hop away from each other and 2-hop away from u are coloring themselves simultaneously. We notice that the distance between these four nodes is maximized when they belong to the circle centered at u and of radius $2R$ and are diametrally opposed. We can compute the distance of two adjacent points denoted v_1 and v_2, we then have $d(v_1,v_2)^2 = d(v_1,u)^2 + d(u,v_2)^2 = 4R^2 + 4R^2 = 8R^2$. Hence $d(v_1,v_2) = 2\sqrt{2}R < 3R$: this contradicts our assumption. $\square$

That is why in the following of this paper, we take $Size\_max\_prio1 = 4$ and $Size\_max\_prio2 = 3$ leading to a smaller bandwidth use.

3.4. Convergence time

As shown in the previous section, the selected setting of the size of max_prio1 and max_prio2 provides the same number of rounds as SERENA, except when the bad scenario occurs. In the bad scenario, the coloring of a node is delayed in OSERENA. Notice that even in this case, the total number of rounds required by OSERENA can still be equal to the total number of rounds required by SERENA. The occurrence of the bad scenario is a necessary but not sufficient condition to increase the number of rounds with OSERENA.

To conclude, the scenario where one OSERENA node u is colored with a delay compared to SERENA happens if the following events occur:

- E_1: $\exists$ v_1, v_2 and v_3, three nodes that are 2-hop away from u and 4-hop away from each other.
- E_2: these three nodes v_1, v_2 and v_3 have a priority higher than u.
- E_3: v_1, v_2 and v_3 are colored simultaneously.

We assume the unit disk graph model of section 2.1.2, including *Assumption A3*. We adopt the following notations. Let $d(u,v)$ denote the euclidian distance between nodes u and v. Let P denote the probability that the bad scenario occurs. We want to estimate an upper bound of this probability. Let P_i denote the probability that the event E_i occurs, with $i \in [1,3]$. We have: $P = P_1 \cdot P_2 \cdot P_3$. For any node u, let $\mathscr{D}(u,R)$, (respectively $\mathscr{C}(u,R)$), denote the disk (respectively the circle) centered at u of radius R. Let $A \setminus B$ denote the set containing exactly the elements of A but not those of B.
The computation of upper bounds of probabilities P_1 and P_2 is done geometrically. On Figure 1, a bound of P_1 corresponds to the probability for v_3 to belong to the hatched area.

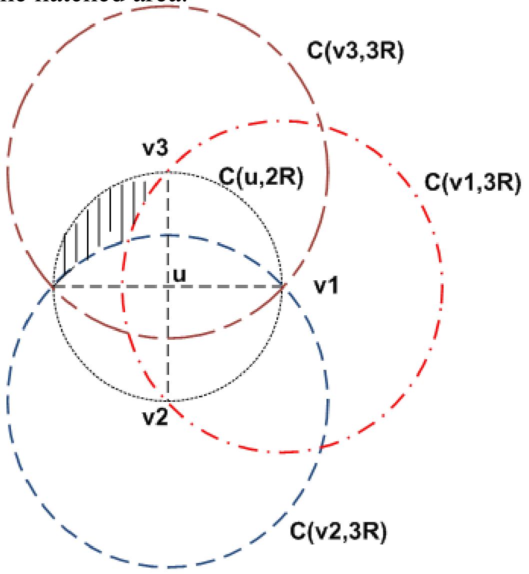

Fig. 1. Possible zone for node v_3.

3.4.1. *Estimation of an upper bound of P_1*

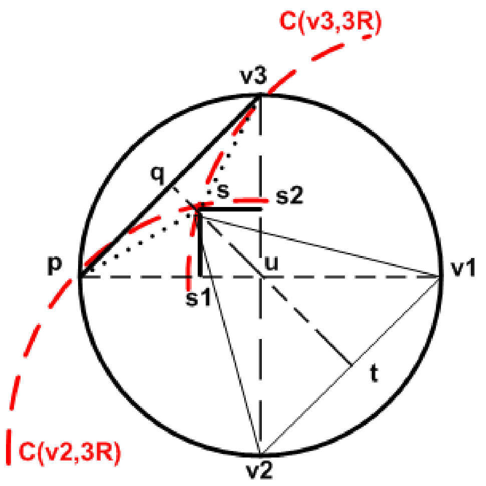

Fig. 2. A figure illustrating the computation of P_1.

The computation of the probability P_1 is illustrated in figure 2. Nodes v_i, for $i \in [1,3]$, should belong to $\mathscr{D}(u,2R) \setminus \mathscr{D}(u,R)$ and should be at a distance belonging to $(3R,4R]$ from each other. To maximize the number of possible nodes v_3, we take $v_1 \in \mathscr{C}(u,2R)$. The choice of v_1 done, we increase the number of possible nodes v_3 by taking $v_2 \in \mathscr{C}(u,2R) \cap \mathscr{C}(v_1,3R)$, approximating $3R+\varepsilon$ by $3R$. We make v_1 and v_2 closer increasing again the possibilities for v_3 by transforming the triangle (v_1,u,v_2) in a right triangle. We then have $d(v_1,v_2) = 2R\sqrt{2}$ computed as the hypotenuse in the triangle (u,v_1,v_2). We now select v_3 that belongs to $\mathscr{D}(u,2R) \setminus \mathscr{D}(u,R) \setminus \mathscr{D}(v_1,2R\sqrt{2}) \setminus \mathscr{D}(v_2,2R\sqrt{2})$, corresponding to the hatched area depicted in Figure 1.

We compute S_P the surface of this area. $S_P \leqslant S_D - 2S_T - S_C$, where S_D is the surface of the disk quarter $\mathscr{D}(u,2R)$, S_T is the surface of the triangle formed by s, u and v_3 and S_C the surface of the square (s,s_1,u,s_2) whose diagonal is y (see figure 1). We first compute $d(u,q)$ in the right triangle (u,q,v_3). We get $2R^2 + d(u,q)^2 = 2^2R^2$. Hence, $d(u,q) = R\sqrt{2}$. In the isocel triangle (v_1,v_2,s), we compute $d(q,t$. We have: $(d(s,t))^2 + 2R^2 = 8R^2$. Since $d(s,t) = d(s,u) + d(u,q)$, we get $d(s,u) = d(s,t) - d(u,q) = (\sqrt{6}-\sqrt{2})R$. We then get:
$S_D = \Pi R^2$ and $S_C = y^2/2 = (4-2\sqrt{3})R^2$.
$S_T = (\sqrt{4-2\sqrt{3}} - 2 + \sqrt{3})R^2$.

We deduce $S_P = (\Pi - 2\sqrt{4-2\sqrt{3}})R^2$.
Hence, $P_1 = \frac{number\ of\ favorable\ cases}{number\ of\ possible\ cases} = \frac{S_P}{4\Pi R^2}$.
Finally, we get $P_1 = \frac{1}{4} - \frac{\sqrt{4-2\sqrt{3}}}{2\Pi}$.

3.4.2. *Estimation of an upper bound of P_2*

For any node u, let us compute P_2 the probability of event E_2: there exists three nodes two-hop away from u with a priority higher than u. This event E_2 can be considered as the intersection of two events E_{21} and E_{22}, where E_{21} means that there exists three nodes in $\mathscr{D}(u,2R)$ with a priority higher than u. Event E_{22} means that three nodes in $\mathscr{D}(u,2R)$ do not belong to $\mathscr{D}(u,R)$. We do not have event $E_2 1$ if and only if in $\mathscr{D}(u,2R)$, 1) u has the highest probability, or 2) u has the second highest probability or 3) u has the third highest probability. Let M denote the number of nodes that are exactly one-hop away from u. The average number of nodes in $\mathscr{D}(u,2R)$ is equal to $4M+1$. We compute P_{21} the probability of event E_{21}. We have $P_{21} = 1 - \frac{3}{4M+1}$.
We can now compute P_{22} the probability of event E_{22}. We get P_{22}=probability that none of these three nodes in $\mathscr{D}(u,2R)$ belong to $\mathscr{D}(u,R)$. Since the nodes are independent, we get $P_{22} = (1 - \frac{\Pi R^2}{4\Pi R^2})^3 = (3/4)^3$. Since events E_{21} and events E_{22} are independent, we get $P_2 = P_{21} \cdot P_{22}$, leading to $P_2 = \frac{27}{64}(1 - \frac{3}{4M+1})$.

3.4.3. *Estimation of an upper bound of P_3*

For any node u, we select the last three nodes v_1, v_2 and v_3, two-hop away from u that color themselves just before u. We want to compute P_3 the probability that event E_3 occurs that is: these three nodes color themselves simultaneously. We can bound P_3 by 1.

3.4.4. *Upper bound for P*

Property 5 *The probability of occurrence of the bad scenario is upper bounded by* $\frac{27}{64}(1 - \frac{3}{4M+1}) \cdot (\frac{1}{4} - \frac{\sqrt{4-2\sqrt{3}}}{2\Pi})$.

Proof. Since $P \leqslant \prod_{i=1}^{3} P_i$, we get $P \leqslant \frac{27}{64}(1 - \frac{3}{4M+1}) \cdot (\frac{1}{4} - \frac{\sqrt{4-2\sqrt{3}}}{2\Pi})$. □

Noticing that $(1 - \frac{3}{4M+1}) \leqslant 1$, a numeric evaluation of the bound yields: $P \leqslant 0.0564$

4. Performance evaluation by simulation

We now evaluate the performance of OSERENA by simulation for various WSNs.

4.1. Simulation modules and parameters

We consider various wireless network configurations, with the unit disk model, where the number of nodes varies from 50 to 200 and the average number of neighbors per node, called density, varies from 8 to 45. We check the connectivity of all the topologies generated by our random topology generator. Three modules are simulated:

- *The Neighborhood Discovery Module* in charge of detecting the creation of new links, testing their symmetry and detecting their breakdown. This is done by means of periodic exchanges of *Hello* messages. The *Hello* message contains the list of addresses of heard/symmetric nodes.
- *The OSERENA Module* in charge of coloring the wireless network, once topology is stabilized.
- *The SERENA Module* used as a reference for a comparative performance evaluation.

We evaluate the number of colors used, the number of rounds needed to color the whole network, the average number of *Color* messages sent per node as well as the average size of these messages. Each result is the average of 10 to 50 simulations.

4.2. Performance results of OSERENA

In this series of simulations, we fix the number of nodes in the interval $[50, 200]$ and vary the node density from 8 to 45. We evaluate the performance criteria of OSERENA and then iterate on another number of nodes.

4.2.1. Number of colors

The main performance criterion of a coloring algorithm is the number of colors needed to color the whole network. This number depends on network topology. First, we want to evaluate the impact of node density and node number on the number of colors used by OSERENA.

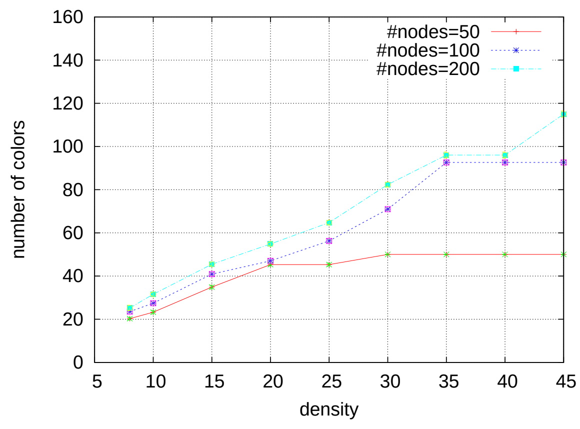

Fig. 3. Number of colors.

The figure 3 shows that the number of colors strongly depends on the density of the graphs, and much less on the number of nodes. Intuitively, the reason is that the color selected by a node, depends only on its 3-hop neighborhood, hence is related to the number of the 3-hop neighbors (which is itself directly proportional to density).

Furthermore, the size of the 3-hop neighborhood is not related to the number of nodes of the graph, hence this last parameter has less impact. This occurs until the transmission range becomes too large and the 3-hop neighborhood includes the whole network (as shown in the figure for a number of nodes $= 50$ and for density $\geqslant$30, where increasing density for a fixed number of nodes is equivalent to increasing transmission range).

4.2.2. Number of rounds

To measure the time complexity of OSERENA, we evaluate the number of rounds needed to color the whole network. More precisely, what is the impact of node density and node number on the number of rounds?

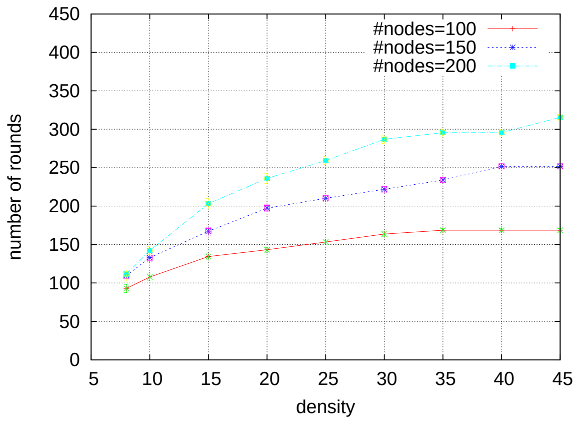

Fig. 4. Number of rounds.

In figure 4, we observe that the number of rounds depends more on the number of nodes in the network than on density.

There is one natural explanation on the observation that the number of nodes has an impact on the number of rounds (and much less on the number of colors, see previous section): in OSERENA, every node u must wait until all the nodes in its $\mathcal{N}(u)$ having a higher priority than itself color themselves. Recursively, each node in this set should do the same. This is likely to lead to waiting "chains", and such chains are longer in larger networks. It contributes to increase coloring delay.

4.2.3. *Number of messages sent per node*

To compute the overhead induced by OSERENA, we first evaluate the average number of messages sent per node for various network configurations, pointing out the influence of node density and node number.

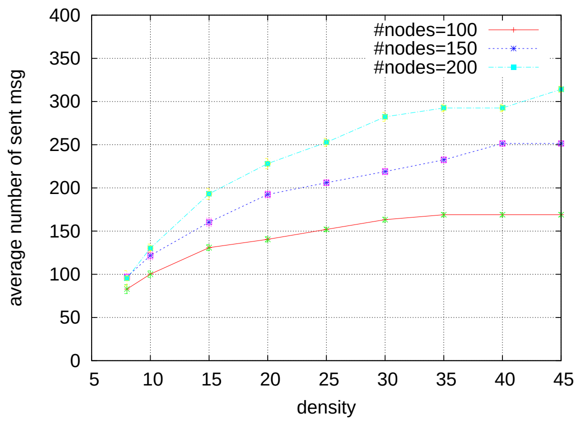

Fig. 5. Average number of messages sent per node.

As illustrated in figure 5, the average number of messages is close to the number of rounds (in figure 4). This is expected since every node sends one message per round until a stopping condition is fullfilled (rule R5): in the simulations, for most nodes, most of the time, rule R5 is not verified.

4.2.4. *Number of bytes sent per node*

Another expression of the message overhead is given by the average number of bytes sent per node for various network configurations. What is the impact of node density and node number on the number of bytes exchanged during the coloring?

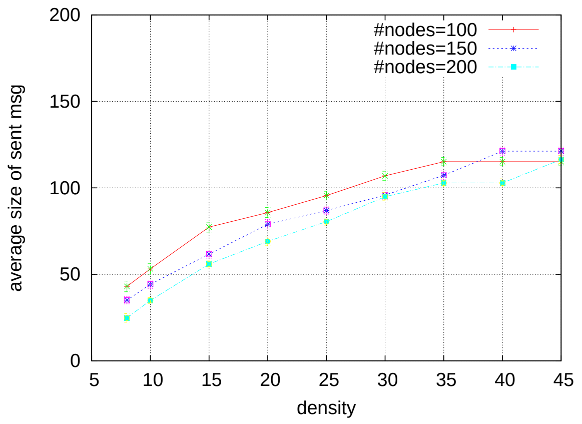

Fig. 6. Average number of bytes sent per node.

From the figure 6, the number of nodes has barely noticeable impact on the average number of bytes sent per nodes, whereas density has a limited, but direct impact. This is a direct consequence of the structure of the *Color* message, which includes 2 bitmaps of colors of the 1-hop and 2-hop neighborhood, which increases linearily with density (e.g. 2 additional bits in message, per additional color in the 3-hop neighborhood).

4.3. Comparison with SERENA

We now compare the performances obtained by OSERENA with those of SERENA. OSERENA ensures that nodes should get the same colors as with SERENA. The open question is at which expense?

4.3.1. Number of colors

Simulation results are compliant with the expected behavior of OSERENA: any node receives the same color with SERENA and OSERENA.

4.3.2. Number of rounds

Simulation results show that even if the bad scenario occurs, OSERENA needs the same number of rounds as SERENA in all the network topologies tested.

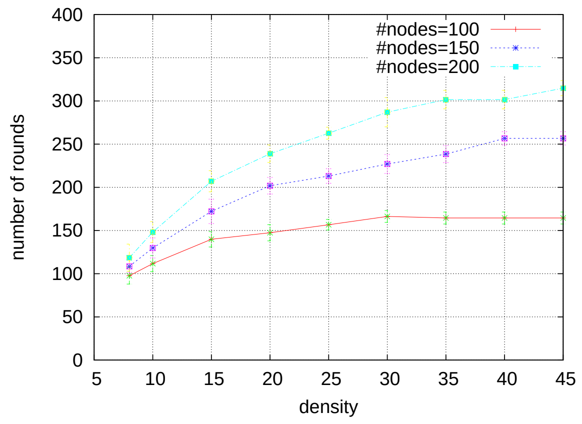

Fig. 7. Number of rounds for SERENA.

Comparing figure 6 and figure 7 we observe that the number of rounds is equivalent. The reason is that the event where OSERENA requires more rounds than SERENA on one node has low probability (see section 3.4.4) ; and then, the occurence of one such event does not automatically increase the total number of rounds for the coloring of the whole network.

4.3.3. Number of messages sent per node

Simulation results show that the average number of messages sent per node is comparable with SERENA and OSERENA for various network configurations. This is a consequence of the previous result.

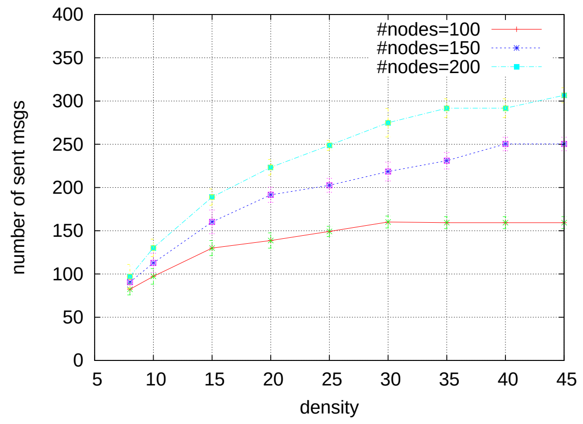

Fig. 8. Average number of messages sent per node with SERENA and OSERENA.

4.3.4. Number of bytes sent per node

Figure 9 depicts the average number of bytes sent per node with SERENA and OSERENA. It points out the benefit brought by the optimization of OSERENA.

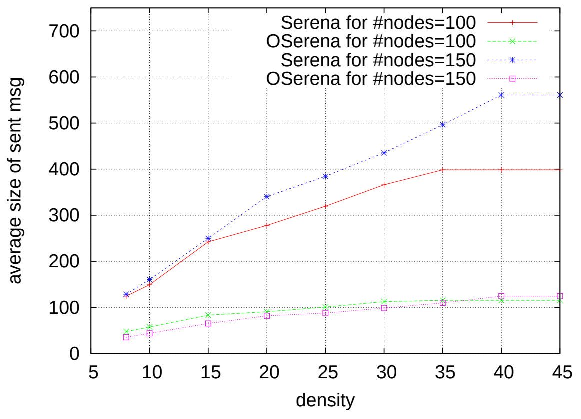

Fig. 9. Average number of bytes sent per node with SERENA and OSERENA.

The figure 9 illustrates the major contribution of OSERENA compared SERENA: the size of the *Color* messages is much smaller.

In SERENA, a *Color* message includes information for each node in its entire 3-hop neighborhood (address, priority, color): several bytes per node in the 3-hop neighborhood. In OSERENA, only a small fixed subset of priorities and addresses of these nodes are exchanged, and only 2 bits per color are required.

Notice that for wireless sensor networks based on 802.15.4, the maximum packet size is 127 bytes, hence SERENA messages are problematic even at the lowest density (and would have probably to be fragmented in several packets), whereas on contrary, OSERENA fits within this limit until high densities.

5. Conclusion

Coloring algorithms have been introduced in WSNs to allow sensor nodes to save energy and bandwidth. Collisions are avoided and nodes can sleep when they are neither sender nor receiver of the transmitted messages. However, their use in dense WSNs is possible only if they are optimized to support such networks. Indeed, the resource constrained nature of sensors combined with the possible high number of neighbors is a real challenge for the design of bandwidth and energy efficient protocols. That is why we have proposed OSERENA, whose performance evaluation results confirm that the WSN is colored with the almost the same number of rounds and exactly the same number of colors as its unoptimized version, but with a message size that does not depend on network density. Consequently, OSERENA enables considerable gains in bandwidth and energy consumption.

References

1. V. Vizing, *On an estimate of the chromatic class of a p-graph*, Diskret. Analiz., 3:23-30, 1964.
2. M. Garey, D. Johnson, *Computers and intractability: a guide to theory of NP-completeness*, W.H. Freeman, San Francisco, California, 1979.
3. S. T. McCormick, *Optimal approximation of sparse Hessians and its equivalence to a graph coloring problem*, Math. Programming, v26, pp. 153-171, 1983.
4. I. Amdouni, P. Minet, C. Adjih, *Node coloring for dense wireless sensor networks*, INRIA Research report RR-7588, http://hal.inria.fr/inria-00582457/PDF/RR-7588.pdf, March 2011.
5. M. Hilgemeier, N. Drechsler, R. Drechsler, *Fast heuristics for the edge coloring of large graphs*, Euromicro Symposium on Digital system design, DSD'03, IEEE computer Society, 2003.
6. P. Galinier, A. Hertz, *A survey of local search methods for graph coloring*, Computers & Operations Research, 3(9), 2547-2562, September 2006.
7. Q. Wu, J.K. Hao, *Coloring large graphs based on independent set extraction*, to appear in Computers and Operations Research, Elsevier, 2011.
8. I. Cargiannis, A. Fishkin, C. Kaklamanis, E. Papaioannou, A tight bound for online coloring of disk graphs, *Theoretical Computer Science*, 384, 2007.
9. B. Clark, C. Colbourn, and D. Johnson, *"Unit disk graphs"*, Discrete Mathematics, Vol. 86, Issues 1-3, Dec. 1990
10. J.C. Bermond, F. Havet, F. Huc, C. Linhares-Sales, *Improper colouring of weighted grid and hexagonal graphs*, Discrete Mathematics, Algorithms and Applications, 2(3):395-411, 2010.
11. V. Rajendran, K. Obraczka, J.J. Garcia-Luna-Aceves, *Energy-efficient, collision-free medium access control for wireless sensor networks*, Sensys'03, Los Angeles, California, November 2003.
12. I. Rhee, A. Warrier, M. Aia, J. Min, *Z-MAC: a hybrid MAC for wireless sensor networks*, SenSys'05, San Diego, California, November 2005.
13. P. Minet, S. Mahfoudh, *SERENA: SchEduling RoutEr Nodes Activity in wireless ad hoc and sensor networks*, IWCMC 2008, IEEE International Wireless

Communications and Mobile Computing Conference, Crete Island, Greece, August 2008.

14. V. Rajendran, J.J. Garcia-Luna-Aceves, K. Obraczka, *Energy-efficient, application-aware medium access for sensor networks*, IEEE MASS 2005, Washington, November 2005.
15. W. Lee, A. Datta, R. Cardell-Oliver, FlexiTP: a flexible-schedule-based TDMA protocol for fault-tolerant and energy-efficient wireless sensor networks, *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, 6, June 2008.
16. S. Gobriel, D. Mosse, R. Cleric, *TDMA-ASAP: sensor network TDMA scheduling with adaptive slot stealing and parallelism*, ICDCS 2009, Montreal, Canada, June 2009.
17. M. Hassan, A. Chickadel, *A review of interference reduction in wireless networks using graph coloring methods*, GRAPH-HOC, 3(1), March 2011.
18. S. Mahfoudh, G. Chalhoub, P. Minet, M. Misson, I. Amdouni, *Node Coloring and Color Conflict Detection in Wireless Sensor Networks*, Future Internet 2010, 2(4), 469-504, October 2010.
19. Ta Xiaoyuan, Mao Guoqiang, B.D.O. Anderson, *On the Probability of K-hop Connection in Wireless Sensor Networks* IEEE Communications Letters, Volume 11 Issue 8, pp 662 - 664, August 2007