Large-scale networked systems: from anarchy to geometric self-structuring

Anne-Marie Kermarrec, Achour Mostefaoui, Michel Raynal, Gilles Trédan, Aline Viana

PUBLICATION
INTERNE
N° 1876

LARGE-SCALE NETWORKED SYSTEMS: FROM ANARCHY TO GEOMETRIC SELF-STRUCTURING

ANNE-MARIE KERMARREC ACHOUR MOSTEFAOUI
MICHEL RAYNAL GILLES TREDAN ALINE VIANA

ISSN 1166-8687

IRISA
CAMPUS UNIVERSITAIRE DE BEAULIEU - 35042 RENNES CEDEX - FRANCE

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES
Campus de Beaulieu – 35042 Rennes Cedex – France
Tél. : (33) 02 99 84 71 00 – Fax : (33) 02 99 84 71 71
http://www.irisa.fr

Large-scale networked systems: from anarchy to geometric self-structuring

Anne-Marie Kermarrec[*] Achour Mostefaoui[**]
Michel Raynal[***] Gilles Tredan[****] Aline Viana[*****]

Systèmes communicants
Projet ASAP

Publication interne n1876 — Décembre 2007 — 19 pages

Abstract: Self-structuring in large-scale networked systems refers to the ability of the participating entities to collaboratively impose a geometric structure to the network. This structure might lead to a geographic organization of nodes or a functional structuring where participating entities get assigned a specific function in the network. Yet, self-structuring is extremely hard to achieve when no global information about the network is available.

In this paper, we present the design and the evaluation of a fully decentralized and robust geometric self-structuring approach relying solely on local observation of neighboring connections for sensor networks. This approach heavily relies on the ability of each node to estimate its position in the network, with respect to other nodes. This refers to assigning virtual coordinates to participating nodes. The contribution of the paper is twofold: *(i)* a simple lightweight and fully decentralized virtual coordinated system (VINCOS) is proposed, relying only on local connectivity information and per-neighbor communication. This is to oppose to most existing approaches relying either on pre-defined positioning referential or signal measurement; *(ii)* the design and evaluation of a network geometric self-structuring approach (NetGeoS) is presented that enables a large set of nodes to configure themselves in arbitrary geometric structures. The evaluation demonstrates that the approach is both efficient and accurate while achieving the geometric structuring.

Key-words: Autonomous system, Geometric structuring, Locality, Self-organizing network, Sensor network, Virtual coordinate system.

(Résumé : tsvp)

[*] INRIA Rennes-Bretagne Atlantique-IRISA, France
[**] Université de Rennes 1-IRISA, France
[***] Université de Rennes 1-IRISA, France
[****] Université de Rennes 1-IRISA, France
[*****] INRIA Saclay - Ile de France sud, France

Centre National de la Recherche Scientifique
(UMR 6074) Université de Rennes 1 – Insa de Rennes

Institut National de Recherche en Informatique
et en Automatique – unité de recherche de Rennes

De l'anarchie à la structuration géométrique

Résumé : Ce rapport présente un système de coordonnés virtuelles adapté à la structuration géométrique des systèmes à grande échelle.

Mots clés : Système autonome, structuration géomt́rique, localité, auto-organisation, réseau de capteurs, système de coordonnées virtuelles.

1 Introduction

Context An *autonomous* system is characterized by the ability to automatically adapt its behavior according to modifications of its environment without requiring an external intervention. This typically falls into the well-known *self-** properties (such as self-healing or self-stabilization, which have proven to be highly beneficial for autonomous systems) [4, 7]. Peer-to-peer systems and wireless sensor networks are typical examples of autonomous systems. In wireless sensor networks, the participating entities (sensors) are deployed over a geographical area, and cooperate in order to collectively accomplish some given function (*e.g.*, consistent gathering of critical information). More generally, sensor-based autonomous systems play an important role in surveillance, data-gathering, or monitoring applications.

Self-structuring represents the ability of a system to let emerge a specific structure, from scratch, without requiring external information. Self-structuring is an important dimension of a system autonomy (especially in terms of scalability issues [19]). In sensor networks for example, self-structuring represents an important requirement for common operations such as forwarding, load balancing, leader election, or energy consumption management. Examples include the partitioning of an area in several zones of equal size for monitoring purposes or the selection of sensors to ensure specific functions for energy saving.

Motivation Obviously, the complexity of a self-structuring mechanism strongly depends on the amount of knowledge initially provided to nodes in the network. At one end of the spectrum, if any node in the system has a complete knowledge of the system, structuring the network is trivial. At the other end of the spectrum, if nodes are only aware of their own neighborhood, ensuring that a given structure emerges from individual decisions is challenging. Let the *external knowledge* be the information provided to a node by an external entity or device. This is to oppose to *intrinsic knowledge* that consists of the information that each entity gathers itself from its observation of the network. The more external knowledge is required, the less robust a system is (especially when deployed in environments where human intervention is difficult). On the other hand, approaches relying mostly on intrinsic knowledge definitively increases system autonomy at the price of a higher communication overhead. This may consequently reduce the system lifetime. In short, the autonomy degree of a networked system is inversely proportional to the external knowledge required to structure the network. It is however crucial to come up with a reasonable trade-off between autonomy and overhead.

This paper presents a robust structuring mechanism, leading to geometric and functional organization that can be deployed from scratch in a networked system where the initial knowledge of each node is limited to its own identity and communication range. Yet, the associated overhead is kept minimum, preserving the lifetime of the system. In the context of this work, we focus on sensor networks. To the best of our knowledge, this is the first *geometric structuring autonomous system* deployed upon those conditions in the literature.

Self-structuring is a powerful tool for wireless networks and a useful operation for getting highly autonomous systems. This is due the fact that it dictates organization laws for nodes in the network. Nodes can then be assigned different adaptive behaviors based on the established organization. In particular, a network organization may follow *geographic* or *functional* principles. Classical examples of a geographic and functional self-structuring are, respectively: a sector-shaped zone clustering for leader election, or an awake and sleep nodes' cluster-based distribution for energy consumption management (see Section 3.3).

This becomes a very challenging goal, as soon as neither positioning referential, boundary delimitation, nor density distribution is provided. Yet, it is crucial in such contexts, that each node be able to infer its position with respect to the other nodes. One solution for this is to allow the nodes to access a coordinate system from which they can obtain a coordinate assignment. Such a coordinate system represents the basic layer on top of which adaptive behaviors can be designed. They consequently lie at the core of autonomous systems design. This constitutes the second contribution presented in the paper.

There is a vast literature in the wireless networks research community containing numerous and valuable proposals for obtaining positioning information [1–3, 5, 6, 8, 10–18, 20]. Yet, most of these approaches rely on some specific assumptions and this shows striking evidence that even more autonomy is required. This has been coped by solutions that use *no position-aware referential points* and that result in *virtual coordinates* being assigned to nodes, instead of geographic coordinates [8, 10, 20]. Virtual coordinates better reflect the real network connectivity, and can consequently provide more robustness in the presence of obstacles. In

addition, since virtual coordinates are relative to nodes' physically close neighborhood, they can tolerate more environment dynamism than absolute geographic coordinates.

Despite having clearly defined outlines and presenting good approximation solutions, previous works on virtual coordinates are computationally- (and message) costly, or hardly practical in wireless sensor networks [8,10]. In addition, the solutions presented in [8,12,20], although not accounting for position-aware landmarks, requires the nomination of well placed entities in the systems to work as anchors or bootstrap beacon nodes.

Instead, this paper proposes a versatile virtual coordinate system for sensor networks. The main difference with related works is that the proposed approach does not rely on any anchors, position-aware landmarks, or signal measurement. Yet, nodes get assigned virtual coordinates in a fully decentralized way. Nodes derive local connectivity information, solely leveraging their per-neighbor communication.

Content of the paper In summary, the contributions of the paper are the following ones.

- A simple, lightweight and fully decentralized, *VIrtual Networked COordinate System* (VINCOS) that achieves a good coordinate assignment.

  Since we target fully autonomous systems, VINCOS does not rely on any information about density, distribution and size of the networked system. The focus is on environments with a large number of nodes that have a *very limited configuration knowledge* (nodes only know that they have distinct identifiers, see Section 2).

- A *Networked Geometric Structuring* approach (NetGeoS) for autonomous systems.

  NetGeoS can be deployed upon any polar coordinate system, without requiring a node to have any knowledge on the network. Here, we show how NetGeoS builds upon VINCOS. Our approach deals with this challenge by providing elegant mechanisms for networked geometric organizations. We show how different geometric structuring can be easily obtained from the virtual coordinates associated with nodes by VINCOS. Moreover, we show that, despite the presence of obstacles, the proposed approach allows a large set of nodes to collaboratively let emerge geometric organizations, in a scalable, simple, and flexible way.

Outline The remainder of this paper is organized as follows. After introducing our system model in Section 2, we present the design rationale of our approach and give an overview of related works in Section 3. The virtual coordinate system (VINCOS) is described in Section 4. In Section 5, we show how this virtual coordinate system can be used to obtain specific geometric structuring. Performance results are presented in Section 6. Finally, Section 7 concludes this paper and discusses future works.

2 System model

As already indicated, we focus in this paper on wireless sensor networks and more specifically on monitoring sensor-based applications. We target application scenarios where the monitored region is of difficult access and/or human intervention is not feasible. This is captured by the following assumptions.

Nodes The system is composed of a (finite) set of N resource-limited wireless devices (sensor nodes) uniformly scattered on a geographical area[1]. Each node gets assigned a unique identifier i. This is the only difference in the initial knowledge of any two nodes. The node identifiers are not necessarily consecutive integers. With respect to any other attribute (*i.e.* computational, memory, energy, and communication capabilities), the nodes can be seen as clones of each other (they are all "equal"). Each node also has a local clock. The set of local clocks is not synchronized. There is an upper bound on the drift rate of the local clocks.

Communication Each node is able to communicate (through broadcasting) within a radio communication range R. Thus, a node i is able to directly communicate wirelessly with a subset of nodes that are reachable

[1] For the sake of clarity, we assume a uniform distribution. Although it is too preliminary to claim that our algorithm converges regardless of the distribution, we conducted some simulations in such settings. The results show that the system converges in presence of normal distribution or topologies that mix different densities' hot spots.

(we refer to this subset as the *neighbors* of the node i). There is an upper bound δ on the message transfer delay between any two neighbor nodes.

Reachability not only depends on geographical distances but also on natural obstacles. In other words, each node i can communicate with another node j provided that j is located within its transmission range R and no obstacle interferes with the communication. We assume *bidirectional* communication and that the density of nodes is such that the resulting communication network is connected (*i.e.*, the network is not partitioned).

Initial knowledge Initially a node only knows its identity, the fact that no two nodes have the same identity, and a parameter d that will define the size (or dimension) of the virtual coordinate space. While a single coordinate space is considered in the following, our proposal can be easily extended to provide nodes with several coordinate spaces, each targeting a specific application and characterized by a specific dimension.

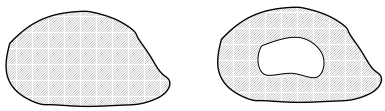


Figure 1: Examples of geographical areas.

Network shape No node is provided with geographical topology information (such as physical obstacles). The presented hereafter approaches relies solely on node connectivity. Yet, a physical obstacle (*e.g.*, a high mountain separating two valleys that merge at some point) can prevent connectivity between nodes that are otherwise geographically close. Examples of geographical areas covered by the nodes, together with their borders, are described in Figure 1. The nodes are located inside the stripped zones. The figure on the left side depicts an area where the nodes are "uniformly" disseminated, without obstacles. The figure on the right side depicts an area with a "centered" obstacle (in that case the border from which the coordinates are computed is the outer border).

3 Design rationale

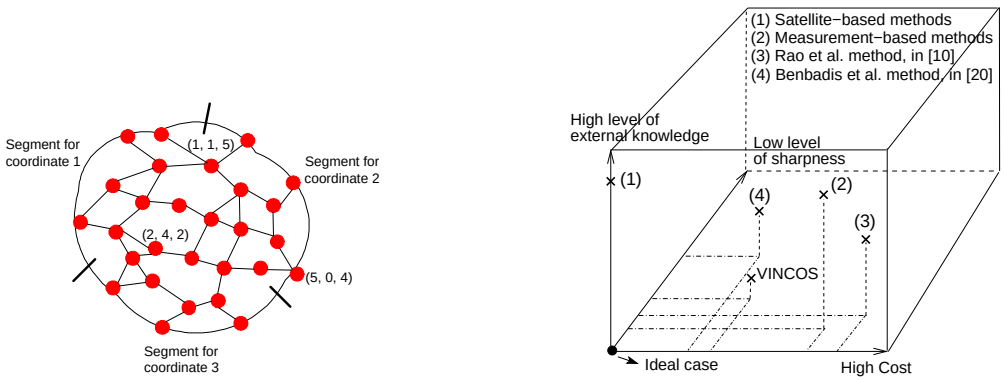
Although being based on extremely simple principles, the approaches presented in the paper implement mechanisms that empower nodes with useful information about the network, such as border detection and cardinality, density cartographic map (or density-based isolines), distance to dense regions, etc. This is achieved using only the natural and local capabilities of wireless communication (*i.e.*, underlying connectivity, messages overhearing, and neighborhood variations). That information can then be used to implement various network functionalities. This section discusses the fundamental principles that underlie each of our contributions.

3.1 VINCOS: a lightweight virtual coordinate system

Position awareness is a key functionality to build and maintain autonomous networked systems. A coordinate system provides each node with a "position" that is both individual and globally consistent. As already observed, a node network awareness may come from two sources: an *intrinsic knowledge* resulting from the algorithm execution (*e.g.*, neighbors set, hop distance to a specific node, etc.), and *external knowledge* supplied by external devices (*e.g.,* satellite or radio signal device), or design hypothesis (such as "all sensors have a unique *id*", "the network is initially connected", etc.).

The physical distance between nodes sharing the same "position" is a relevant metric to estimate a coordinate system's quality [14,20]. Some network functionalities require extremely sharp coordinate system (*e.g.*, greedy routing), while others allows for a weaker form (*e.g.,* data aggregation). The paper refer to a coordinate system quality as its *sharpness*.

In VINCOS, each node is initially configured with the parameter d indicating the dimension of the coordinate system (number of coordinates). The virtual coordinates of a node i are consequently a tuple $(x_1, \ldots, x_d)$, where x_j is the projection of i on the jth axis of the d-dimensional virtual space.



(a) (b)

Figure 2: (a) An example of virtual coordinates. (b) VINCOS w.r.t. cost, efficiency, and external knowledge.

The virtual d-dimensional space is defined as follows. The border of the geographical area covered by the nodes is partitioned into d "segments". This border segments can have the same size or different sizes[2]. Let us consider any axis j, $1 \leq j \leq d$, of the coordinate system. The coordinate x_j is the length, in hops, of a shortest path from the node i to the border segment j (*i.e.*, to the closest node on that border segment). This is illustrated on Figure 2(a) through a simple example. The coordinate system is 3-dimensional ($d = 3$). The virtual coordinates of three nodes are indicated. The coordinates $(2, 4, 2)$ means that the corresponding node is at distance 2 of both the borders 1 and 3, and at distance 4 of the border 2. As mentioned before, such a coordinate system is not related to any anchors or position-aware landmarks. Instead it captures invariance properties associated with connectivity only.

The sharpness of VINCOS thus, depends on an accurate definition of the *border segments*, *i.e.*, the d "segments" dividing the border of the considered geographical area. To define the border segments, VINCOS relies on a *belt* construction mechanism. The resulting belt, defined as a set of *border-belt nodes*[3], is a one-dimensionnnal connected structure that (*i*) enables communication among border-belt nodes along two different paths; (*ii*) allows an order assignment to these nodes; (*iii*) is one-hop wide; and (*iv*) has proportional size wrt the network size[4]. This ensures that, given the broadcast communication pattern, all message forwarded along the belt (complete round) reaches every border-belt node.

Nevertheless, discovering a connected border at a low cost and in an accurate way is difficult. We describe at the Section 4.3, how VINCOS computes such a belt, referred hereafter as *border-belt*.

3.2 Positioning systems: related work

The importance of a coordinate system for autonomous systems is demonstrated by the vast literature on the topic [1–3, 6, 8–18, 20]. Regarding absolute positioning systems, the literature is dominated by the satellite-based methods, like GPS [6], Glonass [1], and Galileo [2]. Equipping all entities with a satellite receiver constitutes the best way to provide them with a very high level of external knowledge, resulting in a very efficient coordinate system (see Figure 2(b)). Unfortunately, this approach suffers from important drawbacks when applied to sensor networks: it is expensive, energy-inefficient, and cannot work when the sensors are deployed in a zone that is out of satellites receiver scopes (as it is the case in some indoor or underground sensor deployment).

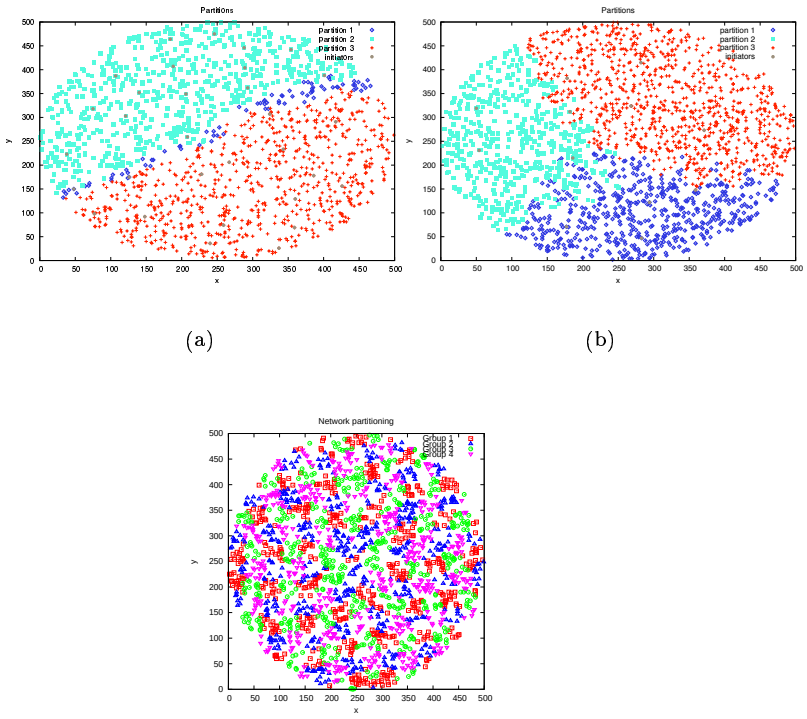
[2] We assume in the paper, w.l.o.g., that they have approximately the same size. This depends only on the algorithm and may easily be changed.

[3] Nodes that are located at the perimeter of the geographical area.

[4] A too small belt could generate inaccuracies in the coordinate system resulting in many non-neighbor nodes being assigned to the same coordinates.

This problem can be solved by equipping only a few entities (called position-aware landmarks) with a satellite receiver, and let the other entities infer their position from connectivity information, or signal strength measurements [3,8,11,13–18]. While allowing systems to be designed with fewer external knowledge (*i.e.*, only few entities know their position), such hybrid approaches are costly and are not efficient from a coordinate ascertainment point of view[5]. Differently, solutions that are based only on connectivity [3, 13, 14] are algorithmically simpler. Nevertheless, they have a high communication cost, being their coordinate system's efficiency strongly dependent of the landmarks density and positioning [12, 20].

A more attractive approach for autonomous systems is when no position-aware referential is used. Clearly, in the absence of external knowledge, self-structuring becomes more complex, requiring systems to become still more autonomous. In order to cope with this issue, recent researches propose solutions that result in *virtual coordinates* being assigned to nodes, instead of geographic coordinates [8, 10, 20].



(a) (b)

(c)

Figure 3: (a)-(b) Example of geometric structuring. (c) Example of functional structuring.

Figure 2(b) visualizes the characteristics of related works in the literature compared to the VINCOS approach. It shows their differences with respect to cost, coordinate system efficiency (*i.e.*, its sharpness), and initial knowledge. The closest to the origin, the better is the compromise between these three criteria.

3.3 NetGeoS: geometric structuring

In the literature, network coordinates are mainly used for routing. We argue that their interest goes far beyond. When appropriately manipulated, network coordinates represents a powerful tool for different kinds of network management. A main contribution of this paper lies in showing how to rely on such a coordinate system for an original purpose.

[5]For example, the signal measurement-based solutions [16, 17] are less efficient in indoor or underground environments besides being algorithmically costless.

A specific network structure should enable nodes to get assigned different behaviors/functionalities depending on their position in the network. Although this can be easily done upon any polar coordinates, we show here how NetGeoS is deployed upon VINCOS.

NetGeoS defines a specific formula that, once applied to the virtual coordinates of each node, gives rise to a specific geometric structuring (that can then dictate organization laws for the nodes in the network, like the one required for: clustering, data aggregation, or energy consumption management). As an example, let us consider sensors disseminated in a large geographical area. These sensors are partitioned in three groups, the *North* group, the *South* group, and the *Equator* group. The Equator group is a simple "straight line" of nodes that separating the north and south groups. Let us consider the application scenario where the nodes in the North and South groups are in charge of collecting some information, subsequently sent (via a routing protocol) to the Equator group. Periodically, a plane flies over the Equator line and collects the relevant data stored in the Equator sensors. To implement such an application, the sensor network should be partitioned in such three groups. It turns out that this can be easily achieved with a 2-dimensional virtual coordinate system. Let (x_1, x_2) be the VINCOS coordinates of node i. i belongs to the North group if $x_1 < x_2$, the South group if $x_1 > x_2$ and to the Equator group if $x_1 = x_2$. Figure 3(a) gives an example of such a geometric structuring in a 2000-node network.

Similarly, in a d-dimensional system, the nodes can be easily associated with distinct partitions as follows: the node i belongs to the partition x, such that $x = \min(x_1, \ldots, x_d)$. The number of partitions depends only on the network connectivity. As shown in Figure 3(b), for a 2000-node network, the simple formula $x = \min(x_1, x_2, x_2)$ defines equivalent partitions, where nodes of each partition are captured by the same property (namely, nodes at distance x from the closest border).

Other geometrical partitioning can be obtained by defining appropriate predicates on the virtual coordinates of the nodes. These predicates may follow *geographic* or *functional* principles. The former defines geometric structuring based on the geographic location of the nodes (as shown in Figures 3(a) and 3(b)). In the later, the geometric structuring is defined based on functional behavior of nodes in the network. More specifically, it takes into account node logical features beyond nodes positioning. For example, consider sensors disseminated in a large geographical area, a functional predicate can allow the distribution of sensors in the network in mixed groups of different states of energy consumption, as shown in Figure 3(c).

Another example of application of a geometric structuring is to define a hierarchical structure to improve protocols which communication cost is $O(N^2)$[6]. Consider, for example, a network where nodes can be divided into p partitions (Figure 3(b) depicts such a partitioning with $p = 3$). An instance of the base protocol can be executed in each partition (*e.g.*, for a leader election) with cost $O\big((N/p)^2\big)$. A protocol which cost is $O(p^2)$ can then be used to piece the results of each partition (*e.g.,* in order to determine a global leader among the leaders of each partition p). This results in a global communication cost of $O\big((N/p)^2 + p^2\big)$, instead of $O(N^2)$ of the base protocol. In fact, it has been already shown in the literature that clustered approaches exhibit better scalability properties than unclustered ones [19].

4 VINCOS: from anarchy to virtual coordinates

To build a meaningful coordinate system, the nodes need to acquire some global and consistent knowledge on the network. The proposed approach provides nodes with a novel and fully decentralized way of acquiring that knowledge. The protocol is composed of four consecutive phases that are described in the following subsections. In order to provide the reader with a global intuition, the underlying general idea is first described.

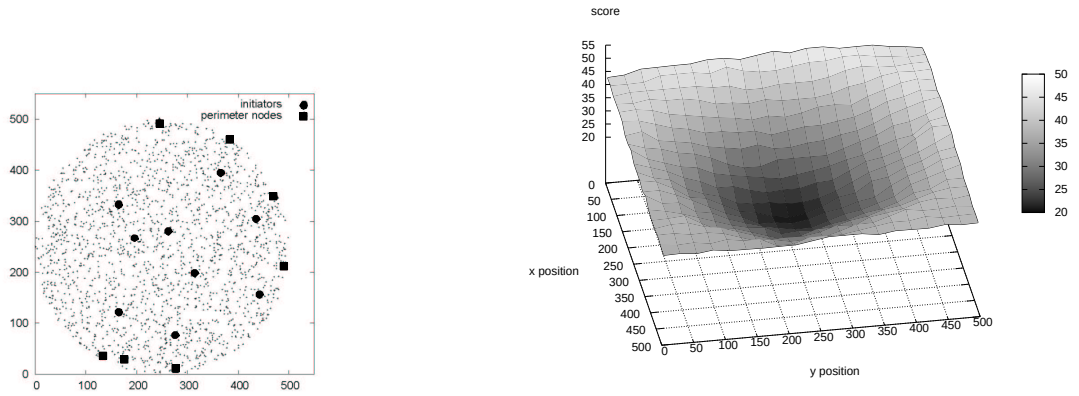
The first phase leverages a density hypothesis to identify a small set of *initiators*, which will have an important role over the next phases and especially in the border detection. During the second phase, each node learns its distance to each initiator[7]. Based on these distances, a belt of border nodes is selected in Phase 3 and the belt gets then divided into d segments (d being the dimension of the coordinate space). Finally, in Phase 4, each process learns its distance with respect to each border segment, and computes its d-dimensional coordinates based on these information. A summary of these phases is described in Table 1.

[6] Being N the number of nodes in the system.

[7] In the following, a distance always refers to the minimal distance between two nodes in terms of number of hops.

We evaluated this protocol through simulations in a large number of networks. We used this set of simulations to setup some specific parameters (more details are provided in the phases' descriptions). For each phase, we also provide a complexity analysis. We measure the convergence time as the number of interactions (an interaction is defined as the time required for broadcasting a 1-hop message). In addition, for the sake of simplicity and w.l.o.g., we assume in the analysis an uniform distribution of nodes in a square grid, or a space divided according to a lattice. In this context, considering N nodes, each row or column of the lattice-divided space contains $\sqrt{N}$ nodes.

4.1 Phase 1: initiator detection



(a) (b)

Figure 4: (a) Nodes with higher scores are located on the border. (b) The perimeter bootstrap nodes and initiators definition for a 2000-node network.

Discovering neighborhood In order to discover its own neighborhood, each node i broadcasts a message carrying its identity. It then waits for a similar message for a fixed period of time (the duration of which is determined from simulation results). That way, each node i learns about its neighbors, and creates consequently a local set $neighbors_i$. Node i then broadcasts the cardinality of this set $|neighbors_i|$ so that each node learns the number of neighbors its neighbors have[8].

Determining initiators At the end of those two communication steps, each node i is able to determine if it is an initiator. An initiator is a local "maximal" node from a density point of view. Intuitively, this is reflected by a node having more neighbors than its neighbors. A node i is an initiator if the following predicate is satisfied:

$$\forall j \in neighbors_i : \quad |neighbors_i| > |neighbors_j|. \tag{1}$$

Simulations on a very large number of scenarios have shown that, for numerous networks where nodes are uniformly distributed, the number of initiators remains low (usually 3, 4 or 5). Although not to be a requirement for the protocol's correctness, a small number of initiators leads to less messages and ensures a faster convergence. Figure 4(a) shows an example of initiators selection.

Leveraging density Since we assume a uniform node density, nodes are expected to have a number of neighbors proportional to the area of the system covered by their radio range. Therefore nodes on the border are expected to have around half of the radio range area *outside* the system and therefore, half less

[8]Remark that only the number of neighbors and not the list of neighbors, is broadcasted.

neighbors. Thus, predicate 1 enables to select initiators that are not on the border (this is important for the next phases, as explained later).

Complexity analysis Clearly, the communication cost of this phase is $2N$ messages, since nodes first inform their 1-hop neighbors about their *id*s and then, once neighbors are discovered, they send a second 1-hop message containing their number of neighbors. The time of convergence is 2 interactions.

4.2 Phase 2: border score definition

The aim of this phase is to provide each node with its *border score*, defined hereafter. To that end, each node determines its distance to each initiator.

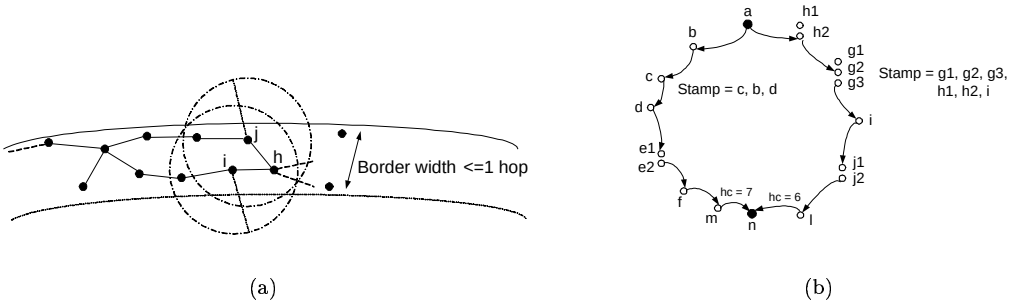
Computing distances to initiators Each initiator i broadcasts a message containing its identity and a hop counter (initialized to 0). After receiving a message m and if m improves its distance to i, node j increases the hop counter and broadcasts it to its own neighbors; otherwise, j discards the message m. The distance of a node j wrt an initiator i ($dist(i,j)$) is defined as follows:

$$dist(i,j) = min\{dist(i,\ell) | \ell \in neighbors_j\} + 1. \tag{2}$$

The border score The value denoted $score_j$ defines the *border score* of a node j, as follows:

$$score_j = \sum_{i \in initiators} dist(i,j). \tag{3}$$

This score definition can be seen as measuring an "average distance" to any initiator. Let $dist(i,\ell)$ be the distance of node ℓ with respect to the initiator i. Let us observe that any node j easily learns $dist(i,\ell)$ for each node $\ell \in neighbors_j$. Thus, each node computes its score and the score of its neighbors. According to Phase 1, the average positioning of initiators is at the center of the system, therefore nodes on the border get a higher score than non-border nodes. Figure 4(b) shows in a gray scale the distance of nodes from the initiators, where the average positioning is at the center.



(a) (b)

Figure 5: (a) Neighborhoods at the two hops i and j intersect, which allows node h not to think he has received probes from different sides. (b) Probe messages carrying the neighbors list of the $2nd$ relaying node

Defining the perimeter The second aim of Phase 2 is to discover some nodes that (*for sure*) are on the border, called of *perimeter bootstrap nodes*. Those nodes are the one initiating border-belt construction at Phase 3. A node j learns if it is a perimeter bootstrap node with respect to an initiator i if the following predicate is satisfied:

$$\forall \ell \in neighbors_j : \quad dist(i,j) \geq dist(i,\ell). \tag{4}$$

Let x be a percentage of the number of initiators. The protocol considers then, that a node belongs for sure to the border and consequently is a perimeter bootstrap node, if the predicate 4 is satisfied for at most

$x\%$ of initiators. Simulation results show that $x = 60\%$ allows detecting a satisfying number of perimeter bootstrap nodes.

Predicate 4 (similar to the one used in [10]), if applied to only one initiator, cannot ensure a correct border detection due to the large amount of *false-positives*[9] it generates. Nevertheless, false-positives are uncorrelated from an initiator to another. Thus, our protocol gets the suppression of false-positives by verifying if that predicate is satisfied for at most $x\%$ of initiators. Figure 4(a) shows an example of perimeter bootstrap nodes' detection.

Complexity analysis This phase requires one message broadcast per initiator. Thus, for N nodes in the system and y initiators, this phase results in a communication cost of $O(y*N)$. Considering nodes uniformly distributed in a lattice-divided space, the time complexity of this phase is proportional to the maximum distance between two nodes in the network area, *i.e.*, proportional to $\sqrt{2N}$.

4.3 Phase 3: border-belt construction

This phase constitutes the heart of the belt construction mechanism. The objective here is to find a list of nodes that all belong to a border-belt, such that the list defines a directed one-dimensional structure. Although being important for obtaining a good sharpness level, a border-belt detection is a complex task if low cost and good accuracy are required.

Probe bootstrap Each perimeter bootstrap node identified at the previous phase sends a *probe* message that will "glue" the border-belt of the system. Figure 7 shows the probe forwarding algorithm.

Probe forwarding Each probe contains two lists of nodes denoted *destinators* and *exclude*. A node i *receives* a probe only if i is in the *destinators* field of the probe (see lines 01-02). At the end of the second phase, each node knows its score, and the score of all its neighboring nodes. When a node i receives a probe message, it selects at most k nodes and forwards the probe to them (k is an application parameter). These k nodes are the nodes with the **highest border score** among a set $candidates_i$ such that:

$$candidates_i = neighbors_i - (destinators \cup exclude).$$

These k nodes define the new nodes of the probe message's *destinators* field. The *exclude* field of this message is then the *destinators* field of the probe message just received by i. This way, probe messages are forwarded by any node in the system. Nevertheless, since they take the path composed by nodes with the highest border score, only nodes on the border will relay probe messages. This also ensures the definition of a one-hop wide border-belt.

In addition to the two previous fields, each probe also contains a *path* field, which describes the nodes the probe message passed through. Each node that relays the probe adds its *id* to this field. Finally, each probe also carries the neighbor list of the $2nd$ relaying node in a field called ***stamp***. This is used to uniquely identify each side of the border-belt. In fact, since nodes in a border-belt are at most one-hop far apart, two probes could take the same portion of the border-belt with different paths (see Figure 5(a)). Therefore, the *path* field is not enough to uniquely identify the side that the probe has taken. Nevertheless, the neighbor lists of two nodes of the same side of the border-belt are always intersecting[10], and therefore constitutes a good stamp. Figure 5(b) shows an example of the border-belt side verification described here before.

Segment definition When a node j receives two probes that have been initiated by the same perimeter bootstrap node i and that contains two different paths and stamps (so, it receives one from its "left" side, the other from its "right" side), node j then considers these two paths as defining the border-belt. Such a node j, called a *segment definer* (see lines 03-06), divides this border-belt in d distinct segments[11]. Finally, each segment definer j sends a *segment message* containing: the set of d segments, the list of border-belt nodes composing each segment, and the *id* of the perimeter bootstrap node that has initiated the corresponding border-belt definition. This message is then forwarded along the defined border-belt. When a border-belt node receives such a message, it learns the segment it belongs to (see Figure 6).

[9] Nodes that falsely believe they are on border.

[10] Thanks to the one hop wide border-belt.

[11] Recall that d is an input parameter of the protocol, that defines the size of the coordinate space.

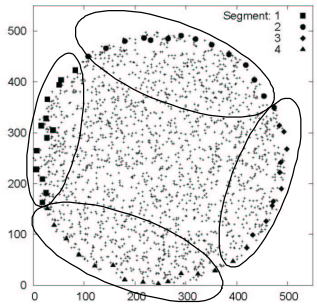


Figure 6: The 4 segments defined for a 2000-node network.

```
upon reception of a PROBE probeIn:
(1)   if ( i ∉ probeIn.destinators )
(2)       return                          # The probe is not for i
(3)   if (probeIn.source ∈ knownProbes_i)
(4)       if (isJunction(probeIn))
(5)           beJunction()                # i is a segment definer node
(6)       return
(7)   knownProbes_i.append(probeIn.source)
      # Relay the probe message
(8)   PROBE probeOut = copy(probeIn)  # copy all fields of probeIn
(9)   let  Candidates_i = neighbors_i - ( probeIn.exclude ∪
          probeIn.destinators )
(10)  probeOut.destinators = selectBest ( Candidates_i )
(11)  probeOut.exclude = probeIn.destinators
(12)  probeOut.path.append(i)
      # Set the stamp if it is not defined
(13)  if ( ( probeOut.stamp = ∅ ) ∧ (probeOut.source ∉ neighbors_i))
(14)      probeOut.stamp = neighbors_i
(15)  send( probeOut)
end
```

Figure 7: The algorithm for the border-belt definition (code for node i).

Preventing ambiguity The fact that the probe bootstrap sub-phase is performed by all perimeter bootstrap nodes may generate several border-belts. Thus, to prevent ambiguity one and only one border-belt has to be considered. To this end, a propagation-extinction mechanism is executed during the segment definition sub-phase. This mechanism ensures that only segment messages containing the smallest perimeter bootstrap nodes' id are forwarded. A node j stores in a local variable sm_j the smallest perimeter bootstrap node's id contained in the received segment messages. The node j only forwards segment messages issued by perimeter bootstrap nodes l whose identity id_l is such that $id_l \leq sm_j$. If $id_l \leq sm_j$, node j also sets sm_j to id_l. Otherwise it discards the message.

Phase	Input	Output
1	*none*	*initiators*: nodes with more neighbors
2	initiators: flood their id	*all*: learn score (average distance to initiators) *perimeter bootstrap nodes*: nodes that are for sure on the border
3*a*.	*perimeter bootstrap nodes*: send probes *highest scoring nodes*: relay the probes	*segment definer node*: node that knows a connected border belt with the probes it received
3.*b*	*segment definer nodes*: defines border segments)	*border-belt nodes*: nodes that own the border-belt
4	*border-belt nodes*: flood their segment number	*all*: learn the small distance to each segment (*i.e.* coordinates)

Table 1: Summary of VINCOS's phases

Complexity analysis The time complexity is proportional to the number of nodes in the belt. As each row or column of the lattice-divided space contains $\sqrt{N}$ nodes, the belt is composed of approximately $4\sqrt{N}$ perimeter bootstrap nodes. Probe messages are forwarded through half of the perimeter in order to complete the border-belt sides' verification. Then, the segment definition sub-phase starts, generating messages to be also forwarded through half of the perimeter. All these operations lower bounds the time of convergence of this phase to $4\sqrt{N}$ interactions.

The communication cost of this phase depends on the number of perimeter bootstrap nodes determined according to the predicate (4). Although it is difficult to precisely estimate, the number of messages exchanged in this phase, this remains small since the perimeter bootstrap nodes represent a small proportion of nodes in the system. So, for z perimeter bootstrap nodes, the communication cost is $z * 4\sqrt{N}$, for a lattice-divided space.

4.4 Phase 4: coordinates definition

Determining coordinates Once a border-belt node has learnt the segment number d it belongs to, it floods the system with a message containing this segment number and a hop counter initialized to 0. Theses messages are then used by all the nodes in the system to compute their shortest distances with respect to the border segment d. This procedure results in each node i being provided with a set of coordinates $(x_1, \ldots, x_d)$ in the d-dimensional space. Section 5 shows how these coordinates can be used to easily define geometric structuring.

Complexity analysis After the segments have been defined, d types of messages propagate the whole network. Each node forwards only one message for each segment d (*i.e.*, the one that comes from the closest border). Thus, Phase 4 has a communication cost proportional to both the number of nodes N in the system and the number of segments d, *i.e.,* $O(d*N)$. Since the d types of messages are simultaneously propagated into the network, the time complexity here corresponds to the time required to travel the maximum distance into the network. Thus, considering the uniform node distribution at the lattice-divided space, the convergence time is $\sqrt{2N}$.

5 From virtual coordinates to geometric structuring

This section deals with geometric structuring, a powerful tool for structuring wireless networks and assigning different functionalities to nodes. We first give a definition of geometric structuring, then describes NetGeoS, relying on VINCOS to appropriately define specific geometric structuring over the network.

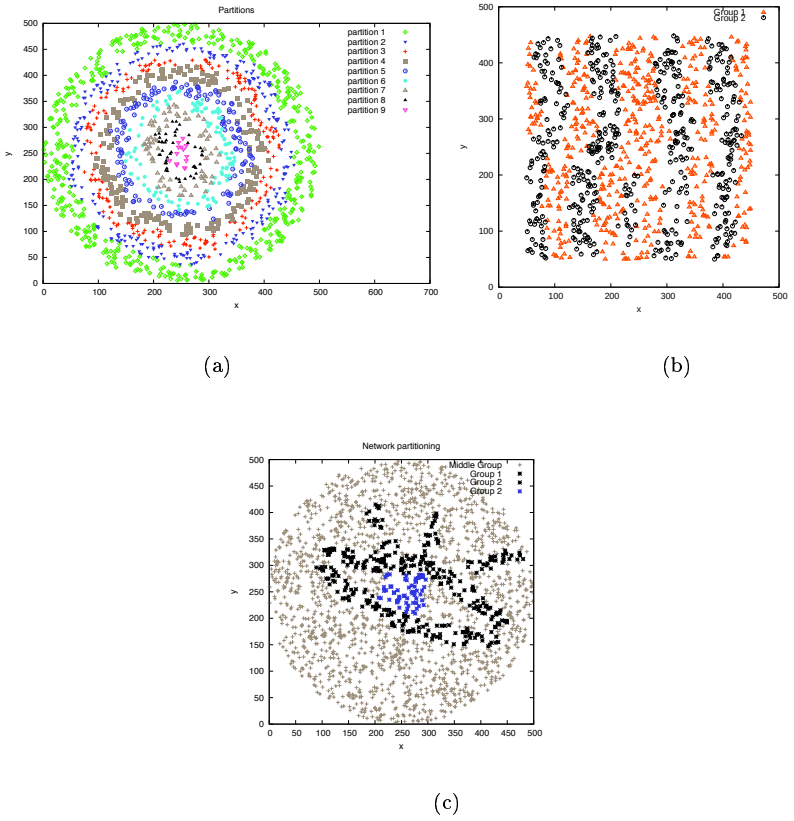
Geometric structuring

Geometric structuring can be defined as a logical partitioning of the network. The aim of NetGeoS is to provide in a fully decentralized way and based only on a local observation of the neighborhood, each node with a partition number. Partitions are then used to fulfill specific applications requirements or systems properties, for example. Formally, let $\mathcal{K}$ be the system nodes coordinate space (in our application, $\mathcal{K} \in \mathbb{N}^d$), and let p be the number of partitions. Let c_i and p_i be the coordinates and the partition number of node i, respectively. Then a geometric structuring function is a function f s.t.

$$\begin{array}{rcl} f : \mathcal{K} & \rightarrow & \{0, \ldots, p\} \\ f(c_i) & \rightarrow & p_i. \end{array}$$

Let us observe that such a definition allows any node to compute the partition number of any other node whose coordinates are known: *each node has a global foresight of the system layout.* The partitioning may take a *geographic* or *functional* form as described below.

Geographic partitioning Geographic partitioning refers to the division of the space into geographical zones for application-dependent purposes. NetGeoS enables to achieve such an accurate geographic portioning. This is detailed in Section 3.3 along some examples. To illustrate our purpose, the following function was used to produce the introductory example, where nodes were structured into *North*, *South*, and *Equator* groups (see Figure 3(a)):



(a) (b)

(c)

Figure 8: (a) Geographic partitioning example. (b)-(c) Functional partitioning examples.

$$\begin{aligned} f : \mathbb{N} * \mathbb{N} &\rightarrow \{1, 2, 3\} \\ f(x_1, x_2) &\rightarrow \begin{cases} 1 & when \quad x_1 > x_2 \\ 2 & when \quad x_1 = x_2 \\ 3 & when \quad x_1 < x_2 \end{cases} \end{aligned}$$

Another useful geographic partitioning is the *target-like partitioning*, as shown in Figure 8(a). This is actually a straightforward structuring to achieve using our coordinate system relying on $d = 1$. In this structure, each node gets as a partition number, its minimum hop distance to the border. This can be a useful structure for tracking application for example, where all the inner rings could be set in sleep mode, the only active partition being the border. Whenever a node from a partition p senses something, it wakes up the partition $p + 1$, so that the network is gradually woken up.

$$\begin{aligned} f : \mathbb{N} &\rightarrow \mathbb{N} \\ f(x_1) &\rightarrow x_1 \end{aligned}$$

For the sake of readability, we have used a few number of partitions in the depicted examples. Nevertheless, NetGeoS allows for any number of partitions, with a fine-grained precision (until one hop wide).

Functional partitioning NetGeoS can also be used to achieve functional partitioning. In geographic structuring, a specific location is associated with each partition. In functional partitioning, each partition is associated with a given application or system-dependent function. For example, one can use this feature to get a structure empowering graph-coloring algorithms, or moreover to apply divide-to-rule methods.

Using the "line" predicate introduced in Section 3.3 (see Figure 3(a)), one can create parallel vertical lines at each j_v jumps, as depicted on Figure 8(b)), using:

$$\begin{array}{rcl} f : \mathbb{N}^4 & \rightarrow & \{0,1\} \\ f(x_1,x_2,x_3,x_4) & \rightarrow & max(x_1,x_3) \mod j_v \end{array}$$

Creating parallel horizontal lines at each j_h jumps can be similarly done by using:

$$\begin{array}{rcl} f : \mathbb{N}^4 & \rightarrow & \{0,1\} \\ f(x_1,x_2,x_3,x_4) & \rightarrow & max(x_2,x_4) \mod j_h. \end{array}$$

Likewise, one can define a regular lattice ($j_h = j_v = 2$) in the following way (see Figure 3(b)):

$$\begin{array}{rcl} f : \mathbb{N}^4 & \rightarrow & \{0,1,2,3\} \\ f(x_1,x_2,x_3,x_4) & \rightarrow & max(x_1,x_3) \mod 2 \\ & & +2 * (max(x_2,x_4) \mod 2) \end{array}$$

A functional partitioning may be used to assign various functionalities to nodes depending on their partition number. For example, the previous described predicate can be used to distribute nodes between awake and sleep states for energy consumption management.

More complicated shapes can be drawn by mixing equations (especially if one knows the size of the network), such as the *eye-like* partitioning depicted in Figure 8(c) and defined from the following equations[12]:

$$\begin{array}{rcl} f : \mathbb{N}^4 & \rightarrow & \{0,1,2,3\} \\ & & \left\{ \begin{array}{lll} 0 & if & eyelid \\ 1 & if & pupil \\ 2 & if & iris \\ 3 & if & eyelashes \\ 4 & otherwise & \end{array} \right. \\ f(x_1,x_2,x_3,x_4) & \rightarrow & \end{array}$$

, where the predicates *eyelid*, *pupil*, *iris*, and *eyelashes* are defined as follows:

Condition	Description
eyelid	$(x_0 = 9 \wedge x_2 < x_0) \vee (x_2 = 9 \wedge x_0 < x_2)$
pupil	$x_1 = x_2 = x_3 = x_4$
iris	$(abs(x_0 - x_2) < 2) \vee (abs(x_1 - x_3) < 2)$
eyelashes	$(x_0 < 12) \wedge (x_1 = x_2 \vee x_2 = x_3 \vee x_1 = x_3)$

Complexity analysis No message exchange is performed for the NetGeoS deployment. Once the VINCOS system is completed, nodes define their partition based on pre-configured geographical or functional partitioning functions. Thus, the communication cost and time of convergence are equal to 0.

6 Performance evaluation

The section describes the experiments we have conducted to assess both the performance and the accuracy of the proposed approach. The experiments have been done using a discrete event simulator implemented in Java. Note that as we are mostly interested in the algorithmic evaluation, our simulator deliberately does not model all the details of a realistic MAC protocol. Instead, we considered a simplified MAC layer where neither messages losses, nor collisions, nor duplications are considered[13].

[12] Note that we introduce the eye-like example to show the power of the partitioning without having a specific application in mind for this structure.

[13] We believe that, even if these issues may cause longer convergence time, they will not affect the correct execution of the proposed algorithms. A detailed study of their impact on our approaches constitutes a future work.

Experimental setup We implemented the system model described in Section 2. The nodes neither crash, nor run out of battery. They strictly follow the protocol "no Byzantine".

Our simulations involve scenarios where the number of nodes varies from 250 to 2600. The nodes are distributed over a 2-dimensional plane, in an area of 500×500 square units. A node range is simulated as a circle area. Radio ranges from 30 to 50 distance units have been used in the simulations. Nodes broadcast `Hello` messages within their radio range, containing information required in each VINCOS's phase.

We conducted experiments under various node distributions in the network (*i.e.*, uniform, normal, multi-centered normal distribution) and various border shapes, like rectangle- and donuts-shaped[14] topologies.

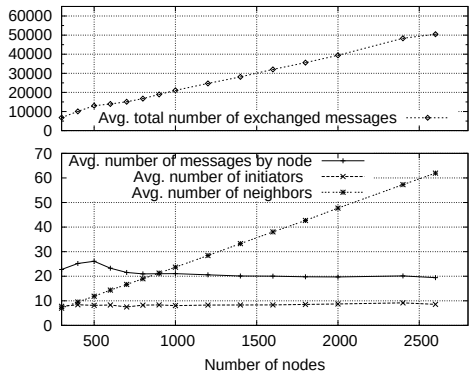


Figure 9: Cost analysis as a function of the network size, for $d = 4$.

Metrics Each point of an experimental curve results from 20 independent experiments. Figure 9 presents the average results of these experiments as the network size increases from 250 to 2600 nodes, for a 40-unit radio range. Network size and radio range are the two main parameters of the simulations since they impact the average number of neighbors, the system size, and the density (which globally impact initiator detection).

Another key parameter is the system shape. This is due the following two reasons. (1) The regularity of the border is important to ensure a correct border detection, (2) VINCOS is tailored for systems with a single border (its outer border).

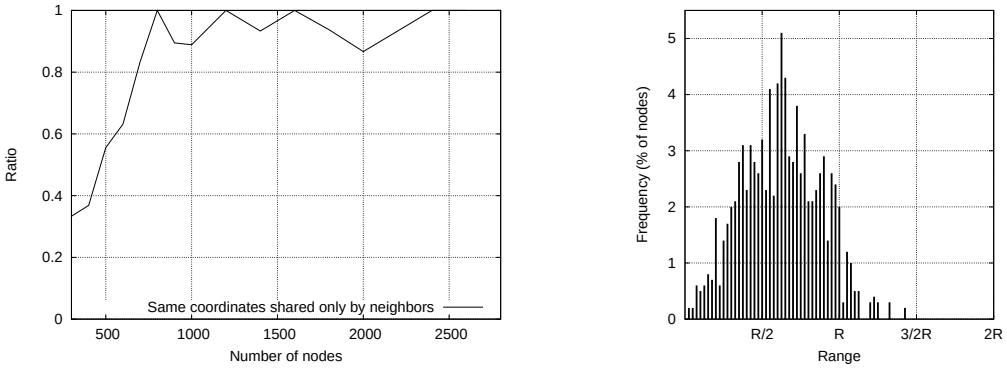
Communication Costs The $O()$ communication cost imposed by each phase during the execution of the VINCOS algorithm has been done on Section 4. Here, is presented the actual communication cost of an execution. This cost sums up the cost of each phase (recall N is the total number of nodes of the system, y and z are the numbers of initiators and perimeter bootstrap nodes, respectively, and d is the size of the coordinate system): $O((2 + y + d) * N + 4 * z\sqrt{N})$.

Figure 9 confirms the theoretical results by showing that the message communication cost per node for the VINCOS construction is low and independent of the network size. It can be observed that for increasing network size and neighbors density, the number of initiators is constant and remains low. In addition, even if the total number of messages exchanged in the network grows nearly linearly with the number of nodes, the average number of messages per node is kept low and presents a slight decrease with the increase of the network size. This can be explained by the fact that inner nodes have a constant number of messages to exchange and that only border-belt nodes exchange additional messages[15]. The number of border-belt nodes, however, decreases with the size of the system, which by consequence, decreases the average cost.

Time of convergence Similarly to communication cost, the theoretical time cost was discussed through the VINCOS's description at the Section 4. Thus, the total expected convergence time is $2 + (2\sqrt{2} + 4)\sqrt{N}$ interactions.

[14] *I.e.*, circle-shaped topologies in presence of voids.

[15] This does not happen in [10], since perimeter nodes flood all the network to discover each other.



(a) (b)

Figure 10: (a) Ratio of neighbors with the same coordinates. (b) Histogram for the distance between two nodes that share the same coordinate.

Sharpness Even though we do not focus on the routing capabilities over VINCOS, it is well known in the literature [20] that accurate coordinate systems lead to correct greedy routing. This is highly dependent on the number of nodes which get identical coordinates as well as their respective physical positions. In order to assess the accuracy of VINCOS, and consequently its routing capabilities, we measured the number of nodes in the system sharing the same coordinates. Among those, we considered the following metric at the granularity of an experiment: *"any two nodes having the same coordinate are neighbors"*. Systems that meet this metric are perfectly suitable for greedy routing: a message reaching the destination's coordinates also reaches the final destination. We then performed 20 independent experiments. Figure 10(a) shows the ratio of experiments that respect the considered metric, for different number of nodes and 50-unit radio range. The figure shows that for some node densities, the considered metric is respected in 100% of the cases. For example, consider the point of the curve $x = 1500$ nodes, it means that in 96% of the experiments, all nodes sharing the same coordinates are neighbors. For networks with a small number of nodes (less then 800 in the figure), the low ratio is explained by the low density in the system.

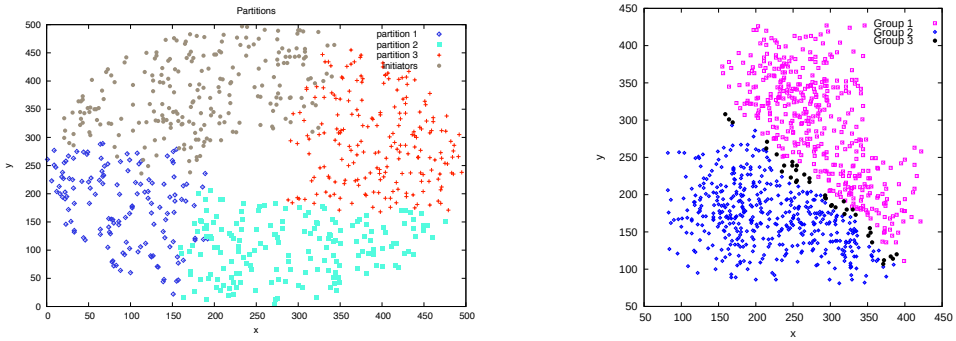
Figure 10(b) shows the distances between two nodes that share the same coordinate in a 1000-node network and 50-unit radio range. We observe that, even though some nodes with the same coordinates are not neighbors – *i.e.*, the distance between them is larger than R in the figure – most of these nodes are neighbors among themselves. More specifically, the graph shows a large concentration of nodes sharing the same coordinates where the average distance between them is less than the ratio range R. These results attest the good level of sharpness of VINCOS's coordinates and its capabilities to support routing.

Different shapes and node distributions As previously discussed, the system shape is a parameter that can strongly impact the coordinate system correctness. In this section, we explore this issue by showing the resulting VINCOS's segment definition under two specific topologies. Figure 11(a) depicts the result for a donut-shaped topology with $d = 4$ in a 2000-node network. We observe that even in the presence of voids (*i.e.*, regions inside the network that do not contain any nodes), VINCOS performs well and correctly defines the required $d = 4$ partitions. The presence of the void leads to the detection of two border-belts, letting the biggest one be used at the virtual coordinate definition (*i.e.*, at Phase 4).

Figure 8(b) shows a functional geometric structuring for a rectangle-shaped topology of 2000 nodes and 40-unit radio range. The obtained geometric structuring shows that VINCOS performs well in a topology with sharp angles too[16].

Similarly, Figure 11(b) shows the partitioning of a 900-node network and 30-unit radio range, where nodes were scattered following a normal distribution. The topology presents three hot spots of different

[16]The study of non-convex or too angle-shaped topologies, like a star, is part of future work.



(a) (b)

Figure 11: (a) VINCOS's segment definition for a topology with a large void in the center. (b) "Line" predicate applied in a topology with three hot spots of different nodes densities.

densities. By applying a "line" predicate, the figure shows the correct partitioning of the network in *North*, *South*, and *Equator* groups.

To summarize, our algorithms works well in the case of network with voids, angle-sharped topology, and different nodes densities: geometric structuring is correctly achieved, which by consequence, indicates that VINCOS is well constructed.

7 Conclusion

This paper has presented (1) the design of an algorithm (*VINCOS*) for assigning virtual coordinates to nodes, and (2) an approach (*NetGeoS*) to structure a network according to geometric patterns. VINCOS protocol not only relies on very weak assumptions (namely, the nodes need only to have distinct and comparable identities), but its design principles are particularly simple. The protocol is fully decentralized and requires neither positioning referential nor signal measurement. Our key contribution in VINCOS is the algorithm for the border-belt definition. This algorithm ensures the low costly construction of an unique regular-shaped and connected belt with only one-hop wide. The upper layer *NetGeoS*, that allows defining appropriate geometric patterns, relies also on simple design principles. Its versatility dimension makes it relevant for defining a functional and geographical partitioning on top of which upper layer scalable services can be implemented.

Simulation results attest of the accuracy of both the network coordinate system as well as the geometric structuring. The structuring is surprisingly well achieved, yet, in a fully decentralized way, and regardless of the shape, distribution of nodes and presence of voids in the network.

Future work includes considering more realistic MAC layer models, considering non-convex irregular shapes and explore geometric structuring to deal with specific application requirements.

References

[1] Glonass positioning system. `www.glonass-center.ru/`

[2] CNES, ESA: Galileo Project. `www.cnes.fr/`

[3] Bischoff R. and Wattenhoffer R., Analyzing Connectivity-Based, Multi-hop Positioning System. *Proc. of IEEE Percom*, pp. 165-176, March 2004.

[4] Dolev S., Self-stabilization. *The MIT Press*, 196 pages, 2000.

[5] He T., Huang C., Blum B., Stankovic J. and Abdelzaher T., Range-free Localization Schemes in Large Scale Sensor Networks. *Proc. ACM Mobicom*, pp. 81-95, September 2003.

[6] Hofmann-Wellenhof B., Lichtenegger H. and Collins J., Global Positioning System: Theory and Practice, *Springer-Verlag*, (5th ed.), 382 pages, 2001.

[7] Pease L., Shostak R. and Lamport L., Reaching Agreement in Presence of Faults. *Journal of the ACM*, 27(2):228-234, 1980.

[8] Moscibroda Th., O'Dell R., Wattenhoffer M. and Wattenhoffer R., Virtual Coordinates for Ad Hoc and Sensor Nertworks. *Proc. of ACM Workshop DIAL-POMC*, pp. 8-16, October 2004.

[9] Nagpal R., Shrobe H. and Bachrach J., Organizing a Global Coordinate System from Local Information on an Ad hoc Sensor Network. *Proc. of IPSN*, pp. 33-348, April 2003.

[10] Rao A., Ratnasamy S., Papadimitriou Ch., Shenker S. and Stoica I., Geographic Routing without Location Information. *Proc. ACM Mobicom*, ACM Press, pp. 96-108, September 2003.

[11] Capkun S., Hamdi M., and Hubaux J.-P., GPS-free positioning in mobile ad hoc networks. *Proc. 34th Annual Hawaii International Conference on System Sciences*, Janvier 2001.

[12] Benbadis F., Obraczka K., Cortès J., and Brandwajn A., Exploring landmark placement strategies for self-organization in wireless sensor networks. *Proc. of IEEE PIMRC*, September 2007.

[13] Benbadis F., Friedman T, Amorim M. D., and Fdida S., GPS-free-free positioning system for sensor networks. *Proc. of WOCN*, pp. 541-545, April 2005.

[14] Caruso A., Chessa S., De S., and Urpi A., GPS free coordinate assignment and routing in wireless sensor netwoks. *Proc. of IEEE Infocom*, pp. 150-160, March 2005.

[15] Shang Y., Ruml W., Zhang Y., and Fromherz M., Localization from mere connectivity. *Proc. of ACM Mobihoc*, pp. 201-212, June 2003.

[16] Niculescu D. and Nath B., Ad Hoc Positioning System (APS) using AoA. *Proc. of IEEE Infocom*, pp. 1734- 1743, March 2003.

[17] Gustafsson F. and Gunnarsson F., Positioning using time-difference of arrival measurements. *Proc. of ICASSP*, pp. VI-553-6 vol.6, April 2003.

[18] Doherty L., Pister K. S. J., and Ghaoui L. El, Convex position estimation in wireless sensor networks. *Proc. of IEEE Infocom*, pp. 1655-1663, April 2001.

[19] Dohler M., Watteyne T., Barthel D., Valois F., and Lu J.-L., Kumar's, Zipf's and Other Laws: How to Structure an Optimum Large-Scale Wireless (Sensor) Network?. *In 13th European Wireless Conference*, April 2007.

[20] Benbadis F., Puig J.-J., Amorim M. D., Chaudet C., Friedman T., and Simplot-Ryl D.", JUMP: Enhancing hop-count positioning in sensor network using multiple coordinates. *Tech. Rep. arXiv cs.NI/0604105*. April 2006.