

# Centroid Virtual Coordinates - A Novel Near-Shortest Path Routing Paradigm

Thomas Watteyne<sup>a,b,\*</sup>, Isabelle Augé-Blum<sup>b</sup>, Mischa Dohler<sup>a</sup>,  
Stéphane Ubéda<sup>b</sup>, Dominique Barthel<sup>a</sup>

<sup>a</sup>*France Telecom R&D, 28 ch. du Vieux Chêne, Meylan, France.*

<sup>b</sup>*ARES INRIA / CITI, INSA-Lyon, 21 av. Capelle, Villeurbanne, France.*

---

## Abstract

Geographic routing has received an increasing attention in the context of Wireless Sensor Networks since it frees the network from the energy-demanding task of building and maintaining a structure. It requires however each node to know its position, which may be a prohibitive assumption for many applications. To this end, some prior work has focused on inferring a node's location from a set of location-aware anchor nodes.

In this work, we free ourselves from positioning techniques and anchor nodes altogether, and introduce and analyze the concept of virtual coordinates. These coordinates are chosen randomly when a node is switched on, and are updated each time the node relays a packet. As this process goes on, the virtual coordinates of the nodes converge to a near-optimal state. When using a greedy geographic approach on top of these coordinates, we show that the number of hops to reach the destination exceeds the shortest path by a few percent only. Moreover, our approach guarantees delivery even when nodes appear/disappear in the network, and under realistic transmission models.

We analytically prove the correctness of our protocol. Moreover, extensive simulations are used to show that our position-free solution outperforms existing geographic protocols – such as Greedy-Face-Greedy (GFG) or Greedy Perimeter Stateless Routing (GPSR) – in terms of energy-efficiency, path length and robustness.

*Key words:* Wireless Sensor Networks, Self-organization, Virtual Coordinates.  
*1991 MSC:* 68M10, 68M12

---

\* This work was partially supported by the French Ministry of Research under contract ARESA ANR-05-RNRT-01703.

\* Corresponding author; thomas.watteyne@ieee.org.

## 1 Introduction and Context

Wireless Sensor Networks (WSNs) have witnessed a tremendous upsurge in recent years, both in academia and industry; this is mainly attributed to their unprecedented operating conditions and a variety of commercially viable applications. Such networks can be used in a wide range of applications, from defense and surveillance to health and intelligent homes [1].

Within this gamut of applications, one federating goal – at least from a networking point of view – is to enable energy-efficient, reliable, and scalable multihop communication. As WSNs can be composed of several thousands of nodes<sup>1</sup> and generally do not enjoy planned roll-outs, it is essential that communication is set up without human intervention. Self-organization can be defined as "*the emergence of system-wide functionality from simple local interactions between individual entities*" [2]. Whereas it may have different definitions depending on the application, for us, this system-wide functionality is routing, which is facilitated by local interactions based on virtual coordinates. Moreover, as routing paths may change (due to node mobility, battery exhaustion, dynamics of the wireless link, etc.) the routing protocol should enjoy self-\* characteristics (self-configuration, self-healing, self-management, etc. [3]).

Arbitrary communication among nodes is commonplace in wireless ad hoc networks, and may be envisioned for in-network communication in WSNs. In most actual WSN deployments, however, data flows are convergecast: data is sent from nodes in the network to a small number of collecting sink nodes. This is verified in the solutions proposed by the industry. Coronis SAS – based in Southern France – attaches wireless sensors to the homes' water meters of a city. Water consumption is reported to a small number of sink nodes which feed the local water supplier's database. Dust Networks – from Hayward, California – focuses on industrial applications. Its highly reliable solution TSMP [4] (parts of which have enjoyed standardization, such as WirelessHART or ISA SP100.11) relies on a single sink node. The convergecast nature of WSNs is listed as a key characteristic of WSNs by the IETF ROLL work group which focuses on identifying routing constraints in the field of WSNs.

For these reasons, we assume a deployment with a small number of sink nodes. Examples of practical applications include:

- an urban WSN reporting the homes' water and power consumption to a small number of sink nodes;

---

<sup>1</sup> In current industrial roll-outs, however, these large networks are cut into independent sub networks of a few hundred nodes reporting to a single sink node.

- a network of fire detectors in a hotel, wirelessly connected to a wireless (broadband) gateway which is capable of contacting the nearest fire brigade;
- a large number of pressure and heat sensors, networked together inside an industrial plant, which deliver sensed data at regular intervals and/or generate alarm messages when measurements exceed a certain threshold; these messages then travel over a multi-hop path to a central controller, linked to a monitoring screen.

We believe that the solution we propose is fit for this class of applications. We propose a scheme where nodes acquire virtual coordinates, and then use these virtual coordinates for geographic routing.

**A Self-Organizing Virtual Coordinate Systems.** We assume nodes do not know their real coordinates. They rather use virtual ones which are not related in any way to their real positions. These coordinates are entirely randomly generated within each node at initialization. In this paper, for simplicity we consider nodes are deployed on a plane, yet it would function equally well for nodes deployed in a three-dimensional space.

With entirely random virtual coordinates, each sent message eventually reaches the destination with the drawback of potentially needing a large number of hops. To mitigate this problem, we propose to update the virtual coordinates of each node on the message's path. This updating process consists of the sending node retrieving the virtual coordinates of its neighbors and updating its virtual coordinates by using a centroid transformation.

**Routing over Virtual Coordinates.** We recently proposed the "3rule" geographic routing protocol [5] which we use on top of the virtual coordinates. As any geographic routing protocol, 3rule stays in greedy mode when possible. If the greedy mode fails, it uses the sequence of already traversed nodes (stored in the packet) as a basis to elect the next hop node among its neighbors. This protocol guarantees delivery on any static connectivity graph.

When used over real coordinates, it presents a hop count equivalent to the Greedy-Face-Greedy (GFG) or Greedy Perimeter Stateless Routing (GPSR) protocols [6]. These two protocols are the most widely adopted geographic routing protocols for WSNs. They were designed by different people at different times, but are equivalent in essence. We will refer to them as GFG/GPSR. The graph planarization technique used in GFG/GPSR (to be detailed in Section 5.5) causes these protocols to fail when the connectivity graph is not a perfect unit disk graph. As this is the case with virtual coordinates, the 3rule routing protocol is used with virtual coordinates [7].

The integration between the virtual coordinates and the routing schemes results in a novel protocol with the following properties:

- no node needs to know its real coordinates;
- this approach **converges to near-shortest path** rapidly;
- it can be made near-**overhead-free** if combined with a suitable MAC protocol;
- it supports having **multiple sinks**;
- the resulting communication architecture is **simple and robust**;
- topological **changes are rapidly absorbed**;
- it **guarantees delivery**, even with a realistic transmission model.

The remainder of this paper is organized as follows. In Section 2, we detail related work on virtual coordinates. We propose our run-time centroid transformation approach in Section 3. We prove that our proposal converges to a near-optimal state in Section 4. Extensive simulations, detailed in Section 5, are used to corroborate its energy-efficiency and robustness. Finally, Section 6 concludes this paper and presents future work.

## 2 Related Work

This section details related self-organization techniques used for WSNs. After presenting self-organization using DHT and clustering (both terms to be defined), we focus on self-organizing coordinate systems.

### 2.1 Self-Organizing Using DHT Concepts

Virtual Ring Routing (**VRR**) [8] applies Distributed Hash Tables (DHT) techniques for routing in WSNs. DHT are a class of decentralized distributed systems that provide a lookup service used in distributed file systems or peer-to-peer file sharing. In VRR, nodes are identified by identifiers which are unique and totally ordered. Nodes are virtually connected together by increasing identifier to form a virtual ring across the network.

Each node maintains a routing table which entries point toward that node's  $r/2$  immediate predecessors and  $r/2$  immediate successors on the ring ( $r$  being an integer value, a priori known by all nodes). VRR sets up forwarding entries to their successors and predecessors along multi-hop physical paths (through other VRR nodes). As a result, a VRR node contains routing table entries that point toward its nearest neighbors in ID space, but also entries for the paths between other pairs of VRR nodes on which it sits.

Routing is done greedily in VRR: a node forwards a packet toward the node in the routing table whose ID is closest to the destination ID in the packet. The

strength about this technique is that nodes keep routes to an average of  $O(\sqrt{n})$  nodes in an  $n$ -node network. A VRR node thus knows about many nodes besides its predecessors and successors, and has accordingly higher probability of "shortcutting" across the ring, rather than walking it linearly. Although a formal proof of the system is considered future work by the authors, presented simulation results are encouraging.

The main disadvantage of this technique is that each time a node (dis)appears, the multi-hop paths traversing that node need to be torn down. To cope with link unreliability, the authors propose to aggressively filter the neighbor tables of the nodes by thresholding link quality. This may result in a disconnected logical graph, although the physical graph stays connected. We consider this lack of flexibility make VRR impractical in most large WSN deployments.

## 2.2 Self-Organization Using Clustering

Clustering is a self-organization principle inherited from research on ad-hoc networks. Clustering groups nodes into clusters, and clusterheads are elected for each cluster. Routing is then performed hierarchically: between cluster members and their clusterheads, and between clusterheads and the sink<sup>2</sup>. A means of interconnecting clusterheads is to construct virtual backbones [10–12].

The Low-Energy Adaptive Clustering Hierarchy protocol (LEACH) [13] is an early and very simple clustering protocol. Depending on a pre-defined probability, nodes elect themselves as clusterhead; other nodes join the closest clusterhead. If this probability is set to 0.1, one can expect on average 10 clusterheads in a 100-node network. The number of clusters grows linearly with the number of nodes, a behavior which may not be desirable (e.g. for complexity reasons). As clusterheads are placed randomly, some non-clusterhead nodes may end up with no clusterhead within communication range. This results in a disconnected graph, although the network is physically connected.

The Hybrid, Energy-Efficient, Distributed clustering approach (HEED) [14] can be seen as an improvement of LEACH in that it guarantees that, after the clusterhead election phase, all nodes are either clusterheads, or neighbors of a clusterhead. It does so by having an election scheme in multiple rounds; the probability that a node declares itself clusterhead increases with the number of rounds. Partly because clusterheads are probabilistically selected based on their residual energy, HEED achieves energy-efficiency. Moreover, clusterheads are distributed more evenly in the network than when using LEACH.

---

<sup>2</sup> Inter-cluster communication may require intermediate "gateway" nodes to relay the messages [9].

In applications such as automated meter reading [15], the network sits idle for most of the time and transmits some bits of data every now and then. In this case, a large portion of messages traversing the network are signaling messages solely used for maintaining the (clustered) structure. Hence, because of the energy consideration, clustering is less appealing for low-throughput WSNs. For these reasons, in this paper, we strictly limit the self-organization mechanism to assigning coordinates to nodes, coordinates which are used by geographic routing protocols to find multi-hop paths.

### 2.3 Self-Organizing Coordinate Systems

Geographic routing protocols are a promising solution for WSNs as coordinates implicitly structure the network (no clusters are needed or maintained, energy consumed during network idle periods is virtually null). Because GPS-like positioning techniques have been reported as being *cost and energy prohibitive for many applications, not sufficiently robust to jamming for military applications, and limited to outdoor applications* [16], protocols have been proposed for nodes to calculate their coordinates.

**Assuming distances between neighbor nodes are known.** This class of self-organizing coordinate systems answers the following problem: *Given a set of nodes with unknown position coordinates, and a mechanism by which a node can estimate its distance to a few nearby (neighbor) nodes, determine the position coordinates of every node via local node-to-node communication.* This is known to be NP-hard in the general case [17].

The Self-Positioning Algorithm (**SPA**) [18] uses the distances between the nodes to calculate the approximate geographical positions of the nodes in two dimensions. Distances between neighbor nodes are assumed to be obtained by external techniques such as received signal strength (RSS), time of arrival (TOA) or angle of arrival (AOA) location techniques [16]. Using beaconing, every node knows its two-hop neighbors and some of the distances between its one-hop and two-hop neighbors. In a first phase, each node elects two of its neighbor nodes to define its local coordinate system. Using basic geometry, each node can position each of its neighbors in its local coordinate system. In a second phase, an election scheme elects the nodes in the Location Reference Group (LCG) which decide upon a network coordinate system. This coordinate system, common to all nodes, is broadcast. In phase 3, each node rotates and inflates its local coordinate system to match the network coordinate system. This results in a coordinate system in which all the nodes are located geographically, but in a (network) coordinate system which may be rotated compared to the geographical coordinate system. Geographic routing protocols used on top of these coordinates yield the same results as when the

nodes knew their true geographical coordinates.

**Assuming distances between neighbor nodes are *not* known.** The Virtual Coordinate Assignment Protocol (**VCap**) [19] does not assume neighbor nodes can measure distances to one another. It functions in two phases. During the first phase, the nodes elect three anchor nodes in a distributed way. The election protocol identifies three nodes which are on three different borders of the network (all nodes can be elected anchor nodes). During a second phase, the newly elected anchor nodes broadcast a message containing a counter incremented at each hop. By listening to these messages, a node  $i$  determines its coordinates as  $\vec{V}_i = \{V_{i1}, V_{i2}, V_{i3}\}$ , where  $V_{ij}$  represents the number of hops separating  $i$  from anchor node  $j$ . Note that these coordinates are not related in any way to geographical positions.

Greedy geographic routing is performed on top of these coordinates by using Euclidean distance. That is, when node  $i$  wants to send a message to node  $m$ , it sends its message to its neighbor  $n$  with smallest Euclidean distance  $D_{nm}$  to node  $m$ , where  $D_{nm} \hat{=} \sqrt{\sum_{i=1}^3 (V_{mi} - V_{ni})^2}$ . Presented simulation results show that this greedy routing scheme yields lower success ratio and higher hop count compared to the same scheme used over geographical coordinates. [20] presents experimental results confirming the practicality of this approach.

Although [21] and [20] are equivalent to VCap, they are less practical as they assume that anchor nodes are placed manually inside the network. In [22], the authors stress the impact of the number and the location of the anchor nodes. [23] shows that, in high density networks<sup>3</sup>, several nodes may end up with the same coordinates, leading to routing ties. Liu *et al.* propose to alter the otherwise integer coordinates in dependency of the node's neighborhood, leading to floating point values. Simulation results show that these floating point coordinates yield better miss ratio and path stretch than when routing on top of true geographic coordinates.

So far, coordinates were considered vectors of distances to anchor nodes. This is not the case in Greedy Embedding Spring Coordinate protocol (**GSpring**) [24] which introduces the spring model. In this model, each link connecting two nodes is considered as a spring. These abstract springs have a rest length which is a function of the node's neighborhood. If two nodes are closer to each other than this rest length (using the non-geographical distance calculated as a function of the nodes' neighborhoods), the repulsion force of the spring causes their 2D coordinates to part away. Inversely, if the length of the abstract spring is larger than its rest length, an attraction force brings the nodes closer.

During initialization of GSpring, an algorithm identifies a predefined number

<sup>3</sup> We call "network density" or simply "density" the average number of neighbors of the nodes in the network.

of nodes at the edge of the network, and initializes their coordinates. The coordinates of these nodes do not change, and they appear as anchors to the spring system. An iterative process causes the abstract springs to be elongated and shortened until the spring system converges. Simulation results show that geographical routing over this coordinate system yields shorter paths than when using the nodes' real coordinates.

In **Rao's solution** [25], the nodes' coordinates iteratively converge using centroid transformation. In the most general case, a first phase identifies the nodes which are on the perimeter of the network. This election phase is made such that perimeter nodes acquire coordinates projected onto a circle while preserving the order of the nodes on the perimeter. Perimeter nodes are considered anchors for the system and do not update their coordinates after the first phase. A second phase is used for the non-perimeter nodes to acquire coordinates. They do so by iteratively exchanging their coordinates, each time updating their own ones by the average value of their neighbors (this is referred to as centroid transformation).

Simulation results show that the success rate of greedy routing with the coordinates obtained by Rao's solution is very close to the success rate of greedy routing using geographic coordinates. Furthermore, in some cases, such as in the presence of obstacles, greedy routing with Rao's coordinates significantly outperforms greedy routing with geographic coordinates. Intuitively, this is because Rao's coordinates reflect the network connectivity instead of the nodes' true positions which are less relevant in the presence of obstacles.

In essence, there are similarities between Rao's solution and the solution proposed in this paper, but we believe that our approach is a significant step forward towards the implementability of WSN routing protocols. Namely, we save energy by omitting the first phase of Rao's protocol, which is the perimeter node determination. Second, we introduce a minimum centroid transformation ring which is vital for implementing this protocol in hardware with finite/limited numerical precision. Furthermore, we mathematically prove convergence and hence universal validity of the approach, whereas Rao only confined to simulations. Finally, as will be shown in this paper, because only the sink nodes behave as reference points<sup>4</sup> (to be explained), the scheme we propose delivers more messages in less hops.

**Hybrid solutions.** Anchor-Free Localization (**AFL**) [26] can be seen as a hybrid solution between VCap and GSpring, combined with the assumption that distances between neighbor nodes are known. It functions in two phases. In the first phase, five reference nodes are elected, four of which are positioned North-South-East-West of the fifth centrally located reference node. As in

---

<sup>4</sup> This is due to the fact that we assume convergecast traffic towards the sink nodes.

VCap, the heuristics use the hop-counts from the chosen reference nodes to approximate the nodes' polar coordinates. In the second phase, a spring model is used and nodes run a force-based relaxation procedure as in GSpring. The difference is that the rest length of the spring between two neighbor nodes is the measured geographical distance. AFL outperforms GSpring because it uses a first phase for the nodes to configure into a topology that resembles a scaled and unfolded version of the true configuration. When force-based relaxation procedure is applied in the second phase, it is less likely to end up in a local minimum.

All solutions presented in this section require an initialization phase to (1) measure the distance to anchor nodes, (2) measure distances between neighbor nodes, (3) iteratively exchange coordinates, or a combination thereof. Besides the fact that it consumes time and energy, the structure built is not robust against topological changes and requires costly periodic rebuilding.

In this paper, we propose to use virtual coordinates which require neither anchor nodes nor initialization phases. The result supports multiple sinks, is robust, and converges rapidly to the near-shortest path. It performs better (both in the energy consumed and robustness) than routing solutions requiring real coordinates, and can be made near-overhead-free if combined with an appropriate MAC protocol.

### 3 Proposing a Self-Organization Scheme for WSNs

#### 3.1 Definitions

To evaluate our proposal, we use path stretch as a primary metric [8]. We call *path stretch* the hop count normalized by the shortest path length (i.e. a path stretch of 1 is optimal). Euclidean distance is used to evaluate proximity.

A connectivity graph is a drawing where nodes are vertices and edges interconnect nodes which are able to communicate. We call *real connectivity graph* the connectivity graph where nodes are placed at their real coordinates. We call *virtual connectivity graph* the connectivity graph with nodes placed at their virtual coordinates. This is the graph the nodes use for routing, although virtual coordinates are not related to real coordinates. Real and virtual connectivity graphs are composed of the same vertices and edges, only the vertices are placed at different positions.

### 3.2 Behavior of the Nodes at Startup

The multihop path followed by the message is determined by the routing protocol, which uses the nodes' virtual coordinates as a basis for its decision. In the following two paragraphs, we describe how nodes initialize their virtual coordinates, considering they do not know anything about the network at startup.

When the nodes are switched on (possibly in an unsynchronized way), each one *randomly* chooses its virtual coordinates. Virtual coordinates are a pair of floating point positive numbers each chosen within  $[0 \dots MaxCoord]$ , where *MaxCoord* is the maximum initial value of a virtual coordinate. The choice of *MaxCoord* is arbitrary and has no impact on the protocol's performance provided it is strictly positive and all nodes use the same. In practice, *MaxCoord* is programmed into all nodes prior to roll-out.

Should the sink have the same behavior, it would need to broadcast its virtual coordinates to all the nodes in the network for them to know where to send their message to. The sink would need to rebroadcast each time it updates its virtual coordinates, which would be very costly. Therefore, the sink node has a slightly different behavior. When it boots, it always chooses virtual coordinates equal to  $[0, 0]$ <sup>5</sup>, and never changes them. As a consequence, nodes always know where to send their messages to, independently from the sink's real position. This is a very interesting behavior, and to our knowledge, the first which never requires the sink to advertise its location.

### 3.3 Run-Time Centroid Transformation

Whenever a node transmits its own or a relayed message, it updates its virtual coordinates according to the transformation we describe in the following paragraphs. As can be expected and gathered from the lower left graph of Fig. 2, the virtual coordinates are random when the nodes are switched on, and the respective initial virtual connectivity graph looks rather erratic. The run-time centroid transformation is used to alter this graph, bringing physically connected neighbor nodes virtually closer to one another.

The run-time centroid transformation is the central part of this paper. While simple, it yields good results as we will see in subsequent sections. Each time a node has a message to send the following is performed:

---

<sup>5</sup> Note that the sink is always *virtually* placed at  $[0, 0]$ , independently from its geographic position.

- (1) The current node retrieves its neighbors' virtual coordinates.
- (2) The current node updates its virtual coordinates with the average of its neighbors' virtual coordinates, i.e. it sets its virtual coordinates at the point of gravity of its neighbors'.
- (3) If after step 2 the current node  $V$  has a neighbor  $W$  virtually closer to him than a threshold distance  $MinVirtual$ , it tells it to update its location according to Eq. (1). This equation calculates node  $W$ 's new virtual coordinates  $W_x^*$  and  $W_y^*$  as a function of its old ones  $W_x$  and  $W_y$ .

$$\begin{cases} W_x^* = \left( \frac{W_x - V_x}{\sqrt{(V_x - W_x)^2 + (V_y - W_y)^2}} \cdot MinVirtual \right) + V_x \\ W_y^* = \left( \frac{W_y - V_y}{\sqrt{(V_x - W_x)^2 + (V_y - W_y)^2}} \cdot MinVirtual \right) + V_y \end{cases} \quad (1)$$

Step 1 is not specific to our solution. Retrieving neighbors' virtual coordinates can be done using a proactive or reactive approach. We assume such a service is available at the node. In Fig. 1 (a), the current node is  $V$ . After step 1, it knows it has three neighbors  $W$ ,  $X$  and  $Y$ , and knows what their virtual coordinates are.

Step 2 *brings neighbor nodes virtually close to each other*. The intuition is that if all nodes do so, a node will get close to its neighbors, which in turn get close to theirs, etc. The resulting emerging behavior is that nodes on a path towards a destination align. Once aligned, a geographical routing protocol follows this path. As we will see in Section 5, this path is near-optimal in terms of number of hops. We call network convergence the transition between the erratic initial and a fully aligned virtual connectivity graph. It is important to note that Step 2 does not need specific messaging between nodes as it is a local calculation performed at the current node. Fig. 1 (b) shows how node  $V$  updates its virtual coordinates with the center of gravity of its neighbors'.

After step 2, and as part of the routing process, the current node sends out the data message towards its neighbor selected by the routing protocol. This packet contains in its header the newly updated virtual coordinates of the current node. As the wireless medium is broadcast by nature, all the current nodes' neighbors hear these new virtual coordinates.

Step 3 has no impact on the fact that the network converges; however, it facilitates implementation on real hardware where virtual coordinates are represented by numbers with limited accuracy. With inaccurate numbers representing the virtual coordinates, nodes virtually close to each other may end up having the same sequence of bits representing their virtual coordinates. If those two nodes are neighbors, it creates ties in the routing process. Avoiding this is quite simple. One has to make sure that neighboring nodes are separated by a virtual distance larger than the accuracy of the binary repre-

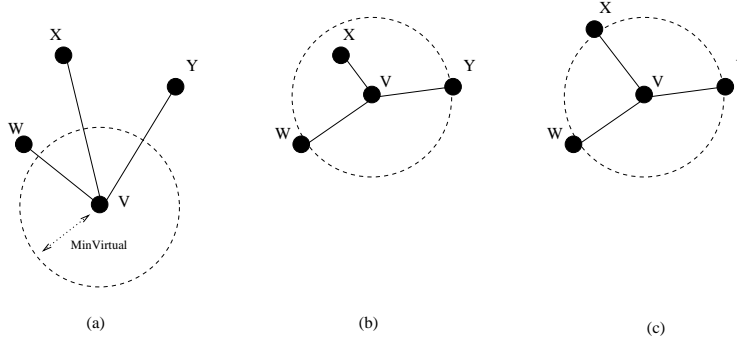


Fig. 1. Run-Time centroid transformation of the virtual coordinates of the nodes. A line interconnects nodes which can physically communicate; a dashed circle represents the points at a distance  $MinVirtual$  from  $V$ .

presentation of the virtual coordinates. We call this distance  $MinVirtual$ . The value of  $MinVirtual$  can be arbitrarily large, and is chosen as a function of the accuracy of the binary representation of the virtual coordinates. The value of  $MinVirtual$  should be positive and all nodes should use the same value. In practice, this is a fixed parameter preprogrammed into the nodes.

Step 3 ensures that neighbor nodes are separated by a virtual distance at least equal to  $MinVirtual$ . This step does not require any communication. All neighbor nodes receive the new virtual coordinates from the current node. All neighbor nodes then check that they are not virtually closer to the current node than  $MinVirtual$ . If this is the case, they apply Eq. (1) to recalculate their own virtual coordinates. Eq. (1) causes the neighbor node which is too close to slide its virtual coordinates on the line formed by the current node and its old virtual coordinates to the position at virtual distance  $MinVirtual$  from the current node. This is represented in Fig. 1 (c): After step 2, node  $X$  ends up too close to  $V$ . Node  $X$  thus updates its virtual coordinates to be at distance  $MinVirtual$  from  $V$ . After updating its virtual coordinates, node  $V$  does **not** need to send them to its neighbors.

### 3.4 Using Multiple Sink Nodes

Multiple sinks may be needed in the network, each one interested in a specific sensed value (light, temperature, etc.). Our solution very elegantly supports the use of multiple sinks. Let's imagine two sinks  $S_1$  and  $S_2$  are deployed in the network. Each node would now have two pairs of virtual coordinates:  $[x_i, y_i]_{S_1}$  for sink  $S_1$  and  $[x_i, y_i]_{S_2}$  for sink  $S_2$ . At initialization, each node randomly chooses both pairs of virtual coordinates. Sink  $S_1$  would use fixed virtual coordinates  $[0, 0]_{S_1}$  (as it is the sink), but randomly chosen  $[x_i, y_i]_{S_2}$  (acting like any other node in the network with respect to sink  $S_2$ ). Sink  $S_2$  would do likewise, i.e. choose randomly  $[x_i, y_i]_{S_1}$  and set deterministically  $[0, 0]_{S_2}$ .

The run-time centroid transformation is applied to both pairs of virtual coordinates by any sending node, regardless of whether the packet is intended for sink  $S_1$  or  $S_2$ . To select the destination sink node, a sending node uses its virtual coordinates  $[x_i, y_i]_{S_1}$  if the packet is intended for sink  $S_1$ , and  $[x_i, y_i]_{S_2}$  if intended for  $S_2$ .

Note that the use of parallel virtual coordinate systems induces overhead as more data needs to be exchanged between neighbor nodes to update the virtual coordinates.

## 4 Proof of Convergence

### 4.1 Witnessing Network Convergence

The three steps presented in Section 3 are used to update the current node’s virtual coordinates. Note that when virtual coordinates are updated, the real position of the node does not change. The main idea of the updating process is to bring neighbor nodes virtually closer. Virtual coordinates are constantly updated when packets flow in the network. In Fig. 2, we show how the virtual connectivity graph evolves while messages are sent through the network. These graphs were obtained by simulation (simulation parameters are detailed in Section 5).

From Fig. 2, we see that the network is converging; from an erratic initial virtual connectivity graph on the left, we obtain an aligned version of the same graph on the right. By formal analysis, we show in the remainder of this section that (1) the network converges to a linear virtual connectivity graph and (2) this linear virtual connectivity graph yields near-optimal paths in terms of number of hops.

### 4.2 Demonstrating Network Convergence

The goal of this subsection is to show that the network converges. The initial virtual connectivity graph is erratic (see Fig. 2, low-left corner), and the nodes potentially fill the entire square of side  $MaxCoord$ . We show that, under the assumption that  $MinVirtual = 0$ , the virtual connectivity graph converges to a linear virtual connectivity graph, i.e. the virtual coordinates of all nodes are on a line. An example of such a completely converged network can be seen in Fig. 3 [center], obtained by simulation. To simplify the analysis, we temporarily consider  $MinVirtual = 0$ , thus removing step 3 from the updating

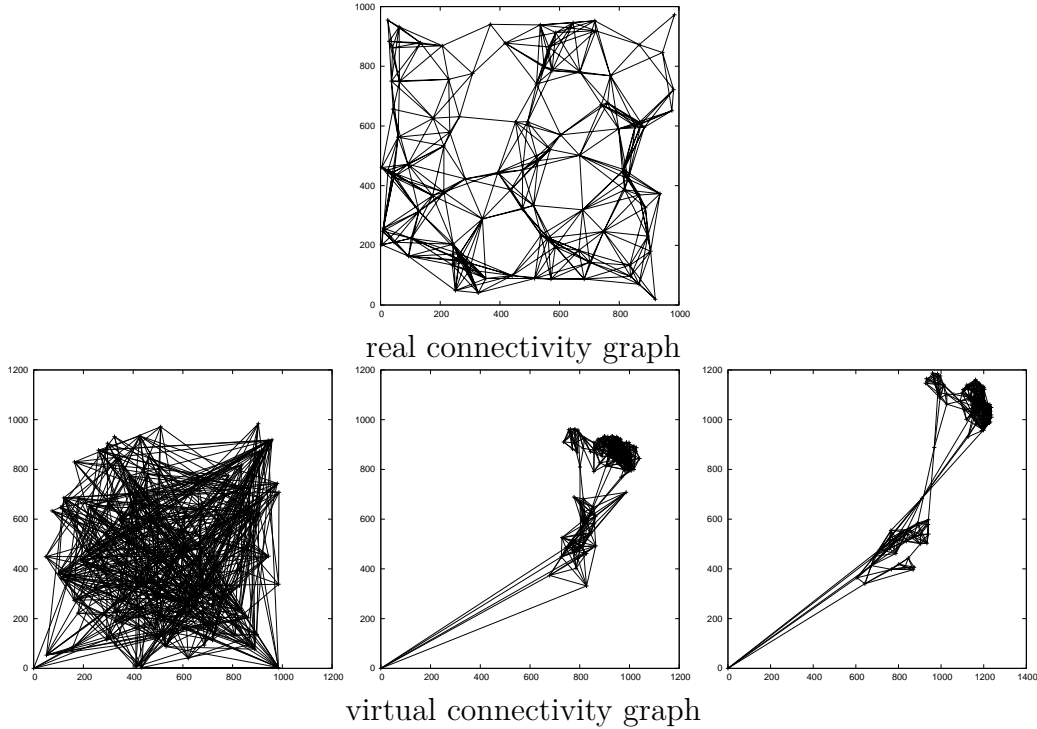


Fig. 2. For the same real connectivity graph (upper part), we draw the virtual connectivity graph after 0, 100 and 500 sent packets (from left to right);  $MinVirtual = 40$ .

process. We consider the network is composed of  $N$  nodes.

We define  $\theta_{min}$  and  $\theta_{max}$  as follows (see Fig. 3 (left)):

$$\begin{cases} \theta_{min} = \min \left( \arctan \left( \frac{y_i}{x_i} \right), 0 < i < N \right) \\ \theta_{max} = \max \left( \arctan \left( \frac{y_i}{x_i} \right), 0 < i < N \right) \end{cases} \quad (2)$$

As depicted in Fig. 3 (left), the lines passing through the sink node and forming angles  $\theta_{min}$  and  $\theta_{max}$  with the lower side of the network form a cone which contains all the nodes in the network.

Let's consider the updating process. In particular, let's see how  $\theta_{min}$  and  $\theta_{max}$  evolve over time. Both values only change if the updating process affects the node defining this angle (in Fig. 3 (left), nodes  $A$  and  $B$  for angles  $\theta_{min}$  and  $\theta_{max}$ , respectively). We will call  $\mathcal{N}_A$  and  $\mathcal{N}_B$  the set of neighbor nodes of  $A$  and  $B$ , respectively.

We will focus on  $\theta_{min}$ , the same analysis applies for  $\theta_{max}$ . By definition, we have  $\frac{y_A}{x_A} \leq \frac{y_i}{x_i} \forall i \in \mathcal{N}_A$ . That is, there is no node other than  $A$  which would decrease  $\theta_{min}$ .

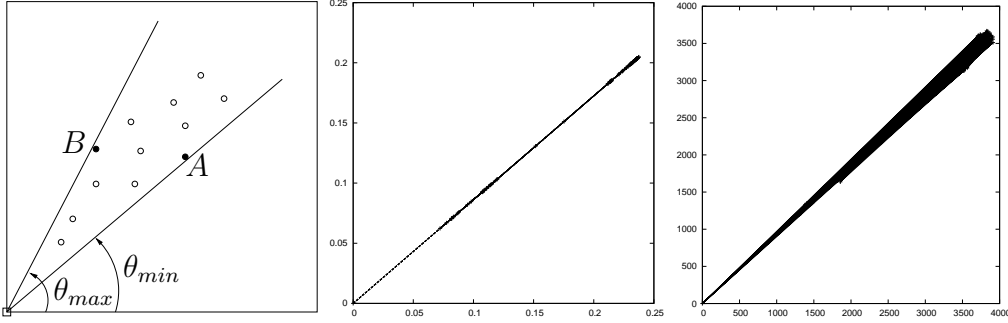


Fig. 3. Definition of angles left]. State of the virtual connectivity graph after 300,000 sent messages for  $MinVirtual = 0$  [center] and  $MinVirtual = 40$  (right).

A node updates its virtual coordinates by the average value of its neighbors'. If the updating node is node  $A$ , it will be shifted closer to the center of the cone, hence increasing  $\theta_{min}$ . We know that  $\tan(\theta_{min}) = \frac{y_A}{x_A}$ . We call  $\theta_{min}^*$  the updated value of  $\theta_{min}$ ; we call  $[x_A^*, y_A^*]$  the updated values of node  $A$ 's virtual coordinates. Remembering that  $\tan(\cdot)$  is a strictly increasing function over  $[0, \frac{\pi}{2}]$ , and that  $\theta_{min}^* \geq \theta_{min}$ , we have

$$\frac{y_A^*}{x_A^*} = \frac{\sum_{i \in \mathcal{N}_A} y_i}{\sum_{i \in \mathcal{N}_A} x_i} \geq \frac{\sum_{i \in \mathcal{N}_A} x_i \frac{y_A}{x_A}}{\sum_{i \in \mathcal{N}_A} x_i} = \frac{y_A}{x_A}. \quad (3)$$

Note that this inequality is not strict because  $\theta_{min}$  is updated only when node  $A$  updates its virtual coordinates.

Therefore,  $\theta_{min}$  increases. A similar analysis shows that  $\theta_{max}$  decreases. We have  $\lim_{m \rightarrow \infty} \theta_{max} - \theta_{min} = 0$ , where  $m$  represents the number of sent messages, assuming all nodes update their coordinates regularly. Network convergence is hence achieved where the network converges to a linear virtual connectivity graph, as depicted in Fig. 3 [center] obtained by simulation.

When  $MinVirtual > 0$ ,  $\theta_{max} - \theta_{min}$  does not strictly converge to zero as neighbor nodes are never virtually closer than  $MinVirtual$ . As shown in Fig. 3 (right), the cone always stays slightly open. Yet after the convergence, nodes are still aligned, and the results of the next subsection apply. Note in Fig. 3 [center] the values of the axis; with  $MinVirtual = 0$ , the network converges to the sink node and each virtual coordinate hence needs to be very accurately represented. This justifies step 3 of the run-time centroid transformation.

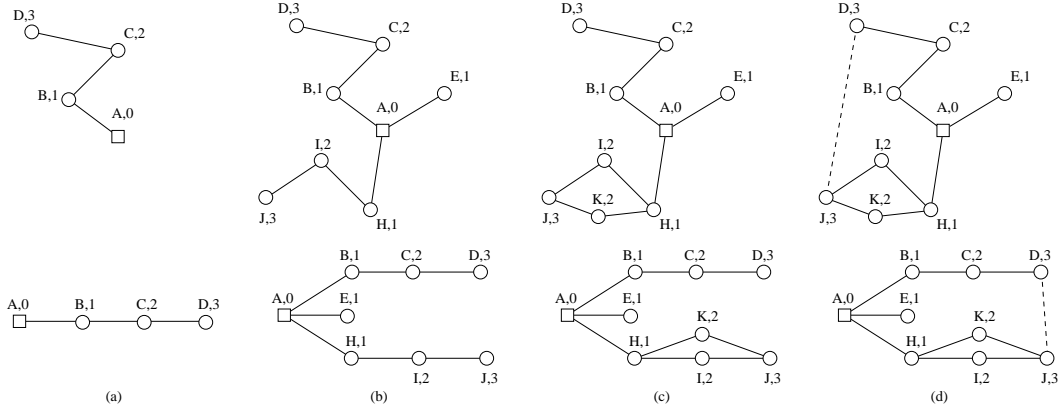


Fig. 4. Demonstrating near-optimality of the resulting path lengths. An identifier ( $A$ ,  $B$ , etc.) and its height are attached to each node. We show both real (upper) and virtual (lower) topologies. The virtual connectivity graphs are linear in reality, but for clarity they are shown expanded vertically.

#### 4.3 Demonstrating Near-Optimality of the Resulting Path Lengths

In the previous subsection, we have shown how the network converges from an erratic to a linear virtual connectivity graph as packets flow through the network. In this subsection, we will show that the nodes' virtual coordinates are ordered properly along this line. Remember that the sink node never updates its virtual coordinates, but rather stays at virtual coordinates  $[0, 0]$ . We call the *height* of a node the minimum number of hops separating this node from the sink (this parameter will be used during the explanation, but is not known by the node). We define *virtual sequence* as the sequence of nodes from virtually close to virtually far from the sink node. We say that a virtual sequence is *correct* if it matches the heights of the nodes. That is, on a given path to the sink, a node with small height should appear before its neighbor with larger height in the virtual sequence. Note that if the virtual sequence is correct, a geographic routing protocol will follow the optimal route in terms of number of hops. The basic principle of a geographic routing protocol is to elect the neighbor node virtually closest to destination as next hop.

Let's start by a **chain topology**, as the one depicted in Fig. 4 (a). After convergence of the virtual connectivity graph, the virtual coordinates of all nodes will be on the same line. We argue that in this case the virtual sequence is correct. This can be easily proven by showing the opposite statement is not possible. Let's consider the virtual sequence is  $A-C-B-D$  ( $B$  and  $C$  have been swapped). As node  $B$ 's neighbors are  $A$  and  $C$ , next time it will update its virtual coordinates, it will appear between  $A$  and  $C$  in the virtual sequence. Moreover, any permutation can be seen as successive permutations between neighbors. In Fig. 4 (a), permuting  $B$  and  $D$ , leads to a virtual sequence  $A-D-C-B$  being equivalent to 3 individual permutations:  $A-B-C-D \rightarrow A-C-B-D \rightarrow$

$A-C-D-B \rightarrow A-D-C-B$ . The proof thus stands for any permutation. Finally, note also that once the virtual sequence is correct, there exists no additional centroid transformation which changes the virtual sequence. The state where the virtual sequence is correct is hence a steady state. With a chain topology, an incorrect virtual sequence is only possible during the convergence of the network, yet in steady state *the virtual sequence is always correct*.

We can extend this analysis to the **star topology** shown in Fig. 4 (b). As there are no links between the branches, there is no effect from the centroid transformations performed on one branch on another branch. Furthermore, since the sink node's virtual coordinates are fixed, the updated virtual coordinates of a node can only affect the virtual coordinates of nodes in the same branch, and cannot propagate to other branches. As the branches are independent, they can be considered as independent chain topologies. Hence, on a star topology, *the virtual sequence is always correct*.

We call **generalized tree** a star topology where several branches can share some of their nodes. Fig. 4 (c) is an example of a generalized tree. A property of this type of topologies is that edges interconnect nodes with a difference of height exactly equal to one. On the virtual connectivity graph, all nodes of height  $h$  are always virtually closer to the sink than neighbor nodes of height  $h+1$ . Let's take the example of node  $I$  and  $K$  in Fig. 4 (c). Independently from whether  $I$  or  $K$  is closer to the sink, the message always follows a shortest path (either  $J-I-H-A$  or  $J-K-H-A$ ).

**Any topology** can be seen as a generalized tree topology with links interconnecting nodes of the same height, as shown in Fig. 4 (d). From those links, virtual connectivity graph incorrectness may arise. Let's take the example of link  $D-J$ . It is possible that  $J$  ends up virtually closer to the sink than  $C$ . In this case, the packet would follow path  $D-J-I-H-A$ , which is one hop longer than the optimal path. *The virtual sequence may be incorrect* in this case.

## 5 Evaluating the Efficiency of the Proposed Approach

We simulated the performance of our scheme with varying network density, number of nodes, propagation models, localization error. We evaluated its performance against the GFG and GPSR routing protocols, assuming nodes running those protocols have perfect knowledge of their geographical position. After presenting the methodology used in Section 5.1, we discuss the assumptions made on the MAC layer in Section 5.2. Section 5.3 focuses on how virtual coordinates converge as messages flow through the network; we measure the impact of network size and density on the converged state and on the speed to reach that state. The robustness of the scheme is shown in Section 5.4 (ro-

bustness against nodes (dis)appearing) and Section 5.5 (robustness against a more realistic propagation model).

### 5.1 *Simulation Methodology*

We wrote a C++ simulator to experiment with, analyze, and visualize the performance and behavior of the different routing schemes<sup>6</sup>. We preferred such a lightweight solution rather than a heavier simulation platform because (1) we wanted to isolate the routing protocols by using an idealized MAC (neither collisions nor transmission delays are taken into account) and (2) we wanted to speed up simulation time to obtain results at high confidence by averaging out many runs.

At each simulation run, nodes are placed randomly in a  $1000 \times 1000$  units square area. We average out the results over large number of runs (typically 5,000). Where appropriate, we depict a 95% confidence interval, which is the interval within which 95% of the individual results fall. The smaller the size of the 95% confidence interval, the more "confidence" one can have in presented averaged results. Messages are sent from randomly chosen connected nodes to a randomly positioned sink node. We use the number of sent messages as a metric of time. Assuming message generation rate is known, it is easy to convert the number of messages back to time.

Hop count is used to evaluate energy efficiency. We implicitly assume that each hop consumes the same amount of energy, independently from the routing protocol used. In this model, energy consumption is not a function of distance between the communicating nodes. This would be a valid assumption for radiated power, but for radios transmitting at a power level below a few mW, their energy consumption is dominated by that of the internal electronic circuits [27]. Lowering the number of hops lowers the energy spent for sending messages, and extends network lifetime.

We compare our solution to GFG/GPSR, the most widely adopted geographic routing protocols, which guarantee delivery in a unit disk graph. To have an unbiased comparison, most of our simulation results are presented in cases where GFG/GPSR give good results. In particular, GFG/GPSR performs best in dense networks (where voids are less frequent).

---

<sup>6</sup> As an online addition to this paper, the source code of our simulator is available at <http://perso.citi.insa-lyon.fr/twatteyn/cvcwatteynecomnet.zip>.

## 5.2 Preliminary Discussion on MAC Layer Integration

Virtual coordinates rely on an iterative convergence process in which a node calculates its new virtual coordinates as a function of its old. For this to function, we assume a system in which:

- a node knows its neighbors and their virtual coordinates;
- all messages have their radio in receive mode when a node sends a data message to hear the node's updated virtual coordinate.

The first assumption is a very general one, shared by the vast majority of routing protocols. Building and maintaining neighbor tables is traditionally done by periodically exchanging `Hello` packets. In an automated water meter reading WSN, nodes sit idle most of the time, reporting water consumption only every 24 hours or so. In this low-throughput context, periodically exchanging `Hello` packets drains most of the network's energy. In this case, it is better to adopt a reactive approach, in which neighborhood is discovered on-demand.

An example of such a MAC protocol is the extended version of 1-hopMAC [28]. We exemplify this protocol in Fig. 5, which represents the radio states of 4 fully-meshed nodes, where node  $S$  is relaying a message. 1-hopMAC extends preamble sampling by adding a series of contention windows in between the preamble and data exchange periods. After hearing node  $S$ ' preamble, nodes  $A$ ,  $B$  and  $C$  send `ACK` messages to inform  $S$  about their virtual coordinates (represented by shaded rectangles). Nodes send an `ACK` message only when the medium is free; if this is not the case, the `ACK` message is sent in a subsequent contention window  $CW$ .  $S$  opens new contention windows by sending a `newCW` message as long as it receives `ACK` messages. After the series of contention windows,  $S$  knows the virtual coordinates of its neighbors, makes a routing decision, and sends its data packet.

Note that any other system which guarantees the requirements listed above can be used. Assuming this is the case, the proposed self-organizing scheme is essentially overhead-free. In terms of memory, geographic routing is known to scale well as the nodes need to store only their own and their neighbors' positions. At constant density, this amount of data is constant regardless of the number of nodes in the network. Because nodes are location-unaware, no specific positioning hardware is needed and our protocol can hence be implemented on standard commercially available nodes. This reduces the cost of the individual sensor nodes. As for computational power, operations are basic. Finally, the impact of communication is negligible. Our virtual coordinate update process re-uses the neighborhood information which is anyway required by the routing protocol. It does require the current node to append its virtual

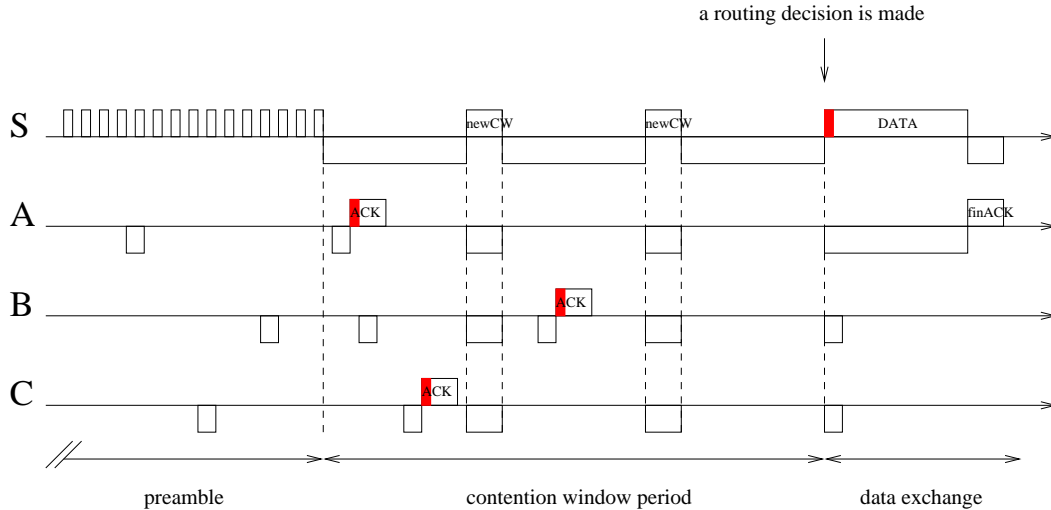


Fig. 5. Extended version of 1-hopMAC [28] which can be used to learn a node’s neighborhood on-demand. The x-axis represents time for 4 fully-meshed nodes  $S$ ,  $A$ ,  $B$  and  $C$ . A rectangle above (resp. under) the axis means that the node’s radio is in transmitting (resp. receiving) mode. No rectangle means the radio is off.

coordinates into the data packet header, but this should not exceed a couple of bytes. We have implemented our system on real motes and confirmed that the energy spent for transmitting and receiving these extra bytes accounts for only 0.45% of a node’s total energy budget.

### 5.3 Convergence of the Virtual Coordinate-Based Solution

Fig. 6 (left) shows how path stretch converges as the number of sent messages increases. Whereas the number of nodes traversed by the first messages is large, the hop count quickly drops and converges to near-optimality. After the cross point at about 80 messages, our system outperforms GFG/GPSR in terms of hop count. Fig. 6 (right) plots the *cumulative* normalized hop count, proportional to the total energy spent since network initialization. After about 300 messages are sent, the proposed solution is more energy efficient than GFG/GPSR. As a numerical example, let us consider an environmental monitoring WSN running for 15 years, with 100 nodes deliver messages reading twice a day; a total of about 1 million messages are sent throughout the network lifetime. Using virtual coordinate routing rather than GFG/GPSR in this setting translates to a significant gain of 61.4% in terms of path stretch, thus in energy.

**Impact of Network Density and Size.** Fig. 7 shows the path stretch obtained after 300 messages have been sent through the network, for GFG/GPSR and 3rule. Fig. 7 (left) shows this for a range of densities in a 100-node network, Fig. 7 (right) for a range of network sizes at a constant density of 5.

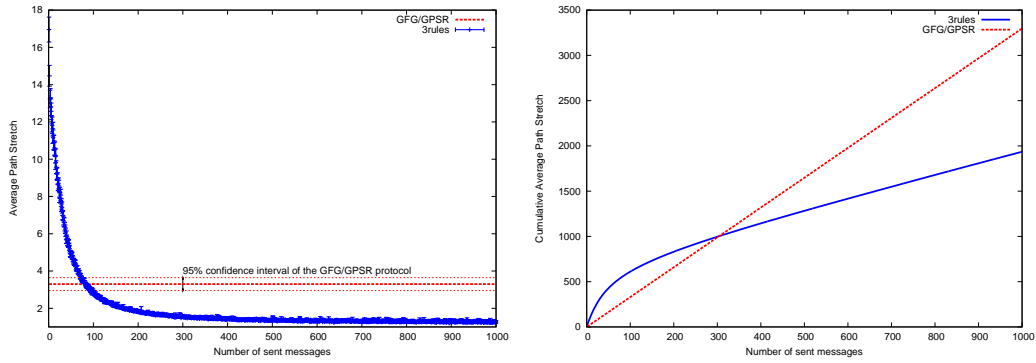


Fig. 6. (left) The average path stretch when using the 3rule and the GFG/GPSR protocols. This graph was obtained for a 200-node network of density 4 and  $MinVirtual = 40$ . (right) This figure is a cumulative version of the figure on the left: it shows the total number of hops consumed by the network since network initialization. The cross point indicates the moment at which our solution is more energy-efficient than the GFG/GPSR protocols.

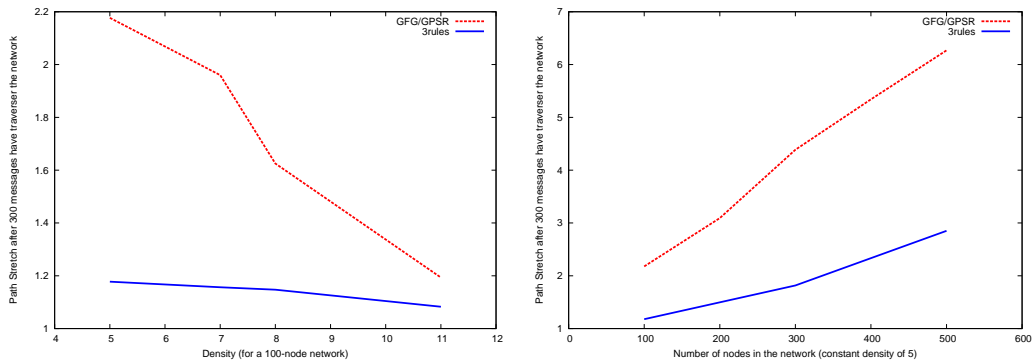


Fig. 7. Path Stretch attained after 300 messages are sent in the network over a range of densities (left) and network sizes (right).

As a general trend, the denser the network, the lower the path stretch. For GFG/GPSR, this is because random networks of higher densities contain fewer void areas which lengthen the paths found. For 3rules, this is due to the fact that updates of the virtual coordinates are done by averaging out the virtual coordinates of more neighbor nodes. This yields a more accurate average value, less prone to routing errors.

The larger the number of nodes in the network, the larger the path stretch. For GFG/GPSR, this is because the probability of encountering geographical dead-ends is larger with longer routes. Similarly, in 3rule, long routes also increase the probability of having the problematic case identified in Section 4.3.

**Speed of Convergence.** As depicted in Fig. 6 (left), the path stretch value of 3rules crosses the one of GFG/GPSR. We plot the number of sent messages when this cross-over happens for different densities and network sizes in Fig. 8. The cross points show when 3rules becomes more efficient, and informs about

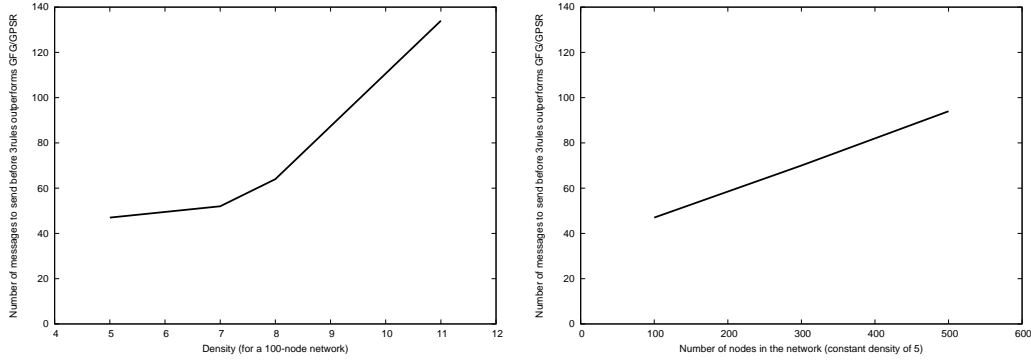


Fig. 8. Cross points of GFG/GPSR with 3rules when varying density (left, the number of nodes in the network is kept to 100) and the number of nodes in the network (right, the density in the network is kept to 5).

the convergence speed. The smaller this value, the faster the convergence.

Convergence is slower for higher densities. This is mostly due to the fact that we define convergence relatively to GFG/GPSR. As shown in Fig. 7 (left), GFG/GPSR are more efficient in high density scenarios, and it takes more messages for the 3rule to outperform GFG/GPSR.

The larger the network at a given density, the longer 3rules takes to cross GFG/GPSR. This is because virtual coordinates arrange radially from the sink node outwards: the sink node virtually attracts its neighbors, which virtually attract theirs, etc. Nevertheless, even in large networks, convergence is fast. As an example, in a 300-node network with nodes transmitting every 5 minutes on average, running 3rules over virtual coordinates outperforms GFG over real coordinates after about 70 seconds, a very small value compared to the total network lifetime.

#### 5.4 *Simulating Node Appearance/Disappearance*

Wireless sensors are unreliable in essence and can disappear (e.g. hardware failure, battery exhaustion, theft, destruction by natural elements) and appear (e.g. additional nodes are rolled out). Communication protocols therefore must present self-reconfiguration, self-healing and self-management characteristics to deal with these changes.

In Fig. 9, we present simulation results when simulating a network deployed in a harsh environment, where a disastrous event (e.g. an earthquake) randomly destroys 30% of all nodes. After that, rescue teams redeploy the same amount of nodes at random locations in the network. In this case (1) many nodes are removed and added, and (2) they do so at the same time.

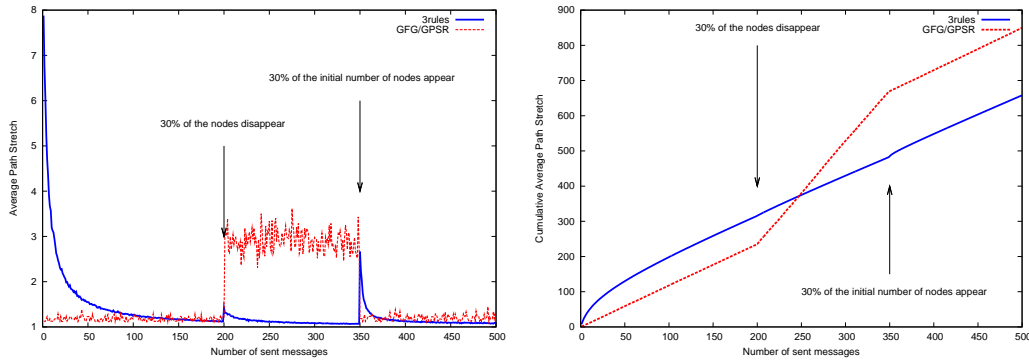


Fig. 9. (left) Path stretch with node appearing/disappearing on a 200-node network. (right) Cumulative hop count shows that our solution quickly outperforms GFG/GPSR.

Note that GFG/GPSR perform worse in sparser networks. This is why the average path stretch of GFG/GPSR evolves stepwise as nodes are added to or removed from the network.

In 3rules, when nodes get disconnected, the other nodes reconfigure themselves by updating their virtual coordinates with a different set of neighbor nodes. As these updated virtual coordinates are used by the node’s neighbors to update their own virtual coordinates, the network reorganizes in a distributed way towards a new converged state. This self-healing process performs similarly when new nodes are added. The network is hence **self-healing**.

This is different from classical ad-hoc routing techniques in which generally, link failures need to be propagated along the routes traversing that link for intermediate nodes to delete those routes, before new routes can be built. During this maintenance procedure, the network can possibly be disrupted and packets can be lost.

From Fig. 9 (right), we see that the topological changes have nearly no impact on the network’s energy consumption when using virtual coordinates. Major topological changes do not significantly affect the expected lifetime.

### 5.5 Performance under a more Realistic Propagation Model

Protocols such as GFG/GPSR use the left-hand rule, over the planar version of the connectivity graph, to recover from geographical dead-ends caused by void area with no nodes. The left-hand rule is used to circumnavigate the void area by traversing the edges surrounding it sequentially clockwise [6]. Planarization consists in removing the edges in the connectivity graph which cross. This can be done by locally applying the Gabriel Graph transformation. Following this transformation, an edge  $[VW]$  is part of the Gabriel Graph *iff* there is no node

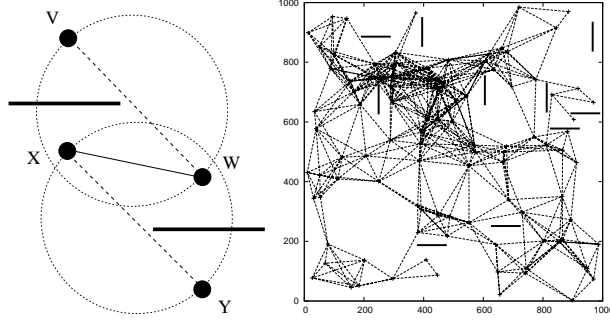


Fig. 10. (left) The two obstacles presented as thick lines cause the planar graph to be disconnected, and the GFG/GPSR protocols to fail. (right) a real connectivity graph with 10 obstacles.

within the disk of diameter  $[VW]$ . This implies communication regions are perfectly circular disks. In real systems, this does not hold and these protocols fail dramatically [29]. To show our proposal holds even when communication regions are not perfectly circular disks, we present two simulation results.

The **first result** is obtained when placing obstacles in the deployment area. The position of these walls of length 100 is chosen randomly and uniformly inside the deployment area; their orientation (horizontal or vertical) is also random. Communication between nodes is possible *iff* the straight line between two nodes does not cross an obstacle (see Fig. 10 (right)).

Fig. 10 (left) depicts a 4-node network with two obstacles; edges are drawn between nodes able to communicate. When node  $V$  executes the Gabriel Graph transformation, it sees no neighbor node within the circle of diameter  $[VW]$ , and decides to keep this edge. The same stands for node  $Y$  keeping the edge  $[XY]$ . Now consider node  $W$ . It will not keep edge  $[VW]$  because node  $X$  is in the circle of diameter  $[VW]$ . Similarly, node  $X$  will remove the edge  $[XY]$ . This results in a disconnected graph. The interested reader can easily repeat this transformation now removing the obstacles, and will realize all nodes are connected (only edges  $[VX]$ ,  $[XW]$  and  $[WY]$  are kept). The fact that the Gabriel Graph transformation is entirely based on a perfect unit disk graph assumption makes it not suitable for real-world propagation conditions.

The individual case depicted in the previous paragraph appears more often when the number of obstacles in the network grows. Hence, as depicted in Fig. 11 (left), the delivery ratio drops with number of obstacles when using GFG/GPSR. In contrast, our system keeps delivering all sent messages.

The **second result** is obtained when localization accuracy is not perfect. So far, the performances of GFG/GPSR were extracted assuming the nodes had perfect knowledge of their position. Yet, any localization technology only offers limited positioning accuracy to the nodes (e.g. around 10m for GPS). Similar to the previous case, the Gabriel Graph transformation used in GFG/GPSR

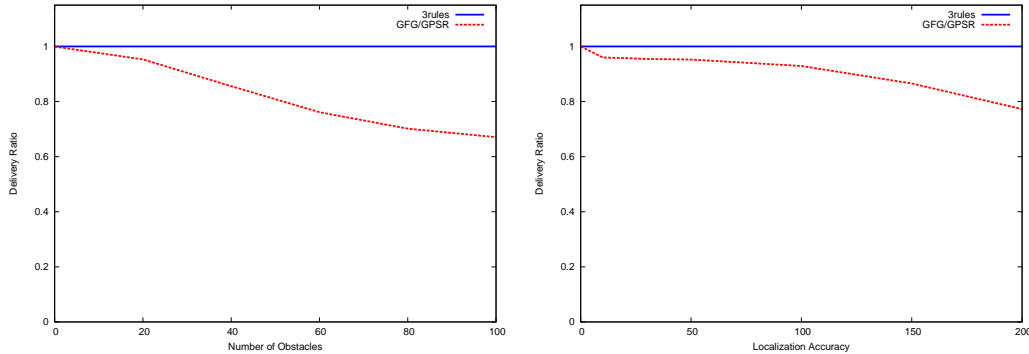


Fig. 11. Simulating the real world: delivery ratio in the presence of obstacles (left) and with limited location precision (right).

does not function with limited positioning accuracy (interested readers are referred to [7]).

By simulation, we mimic a localization technique by informing the nodes about their real location plus/minus a value randomly and uniformly chosen within  $[-accuracy, +accuracy]$ . Fig. 11 (right) shows how the delivery ratio drops with the position accuracy when using GFG/GPSR. As our system does not assume location-aware nodes, it keeps delivering all sent messages.

## 6 Concluding Remarks and Future Work

In this paper, we proposed a communication architecture which uses a geographic routing protocol on top of virtual coordinates. These coordinates are not related to the real coordinates of the nodes, freeing the system from expensive solutions such as GPS. At each hop, the current node updates its virtual coordinates using the centroid transformation. This transformation causes the virtual connectivity graph to converge. Once converged, messages which are sent greedily using these virtual coordinates travel for a number of hops which exceeds the shortest path by only 4% for given system assumptions.

We have proven the convergence of the system to a near-optimal state. Extensive simulations were used to compare performances with GFG/GPSR. We have shown that, despite the fact that our nodes do not need any positioning hardware or protocol, it outperforms GFG/GPSR in terms of energy efficiency. We also show that our solution guarantees delivery even when the number of nodes changes, or when the transmission model is not a perfect unit disk graph.

Our solution seems very well suited for Wireless Sensor Networks. It does not maintain any structure when the network is idle, leading to virtually no

energy consumption during these – possibly long – periods. Furthermore, it supports multiple sinks, and it is extremely robust to nodes appearing and disappearing and irregular communication ranges. The network naturally and quickly stabilizes to a near-optimal state in terms of number of hops. The solution’s complexity is low and constant as the number of nodes in the network increases. This is particularly welcome in networks with a large number of cheap and unreliable wireless nodes.

This paper opens a lot of exciting potentials for research. Throughout this paper, we have implicitly assumed that a routing protocol guaranteeing shortest path is optimal. Whilst being a fair assumption in many cases, it would be interesting to extend this approach to minimum energy paths, and see the impact on the network lifetime. Moreover, the linear virtual connectivity graph to which we converge could be used as a means of guaranteeing QoS parameters such as delay. In its current state, our solution requires a different set of virtual coordinates for each sink node in the network. While the associated overhead is low for a small number of sink nodes, it may become significant in an application with many sinks. In this case, our solution could be extended by having geographically close sink nodes share a common virtual coordinate set.

## Acknowledgments

We would like to thank the anonymous reviewers for their very valuable comments which have helped improve the quality of this paper.

## References

- [1] I. F. Akyildiz and I. Kasimoglu, “Wireless sensor and actor networks: research challenges,” in *International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*. Fort Lauderdale, Florida, USA: IEEE, December 2004.
- [2] F. Dressler, *Self-Organization in Sensor and Actor Networks*, F. Dressler, Ed. John Wiley & Sons, Inc., Hoboken, New Jersey, November 2007.
- [3] J. A. Stankovic, “Research challenges for wireless sensor networks,” *ACM SIGBED Review: Special Issue on Embedded Sensor Networks and Wireless Computing*, vol. 1, no. 2, pp. 1–4, July 2004.
- [4] K. Pister, “TSMP: Time synchronized mesh protocol,” 2008, submitted.
- [5] T. Watteyne, I. Augé-Blum, M. Dohler, and D. Barthel, “Geographic forwarding in wireless sensor networks with loose position-awareness,” in

*18th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. Athens, Greece: IEEE, September 3-7 2007.

- [6] H. Frey and I. Stojmenovic, "On delivery guarantees of face and combined greedy-face routing algorithms in ad hoc and sensor networks," in *Twelfth ACM Annual International Conference on Mobile Computing and Networking (MOBICOM)*. Los Angeles, CA, USA: ACM, September 23-29 2006, pp. 390–401.
- [7] T. Watteyne, D. Simplot-Ryl, I. Augé-Blum, and M. Dohler, "On using virtual coordinates for routing in the context of wireless sensor networks," in *18th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. Athens, Greece: IEEE, September 3-7 2007.
- [8] M. Caesar, M. Castro, E. B. Nightingale, G. O'Shea, and A. Rowstron, "Virtual ring routing: Network routing inspired by dhds," in *SIGCOMM*. Pisa, Italy: ACM, 11-15 September 2006, pp. 351–362.
- [9] O. Younis, M. Krunz, and S. Ramasubramanian, "Node clustering in wireless sensor networks: Recent developments and deployment challenges," *IEEE Network*, vol. 20, no. 3, pp. 20–25, May/June 2006.
- [10] F. Théoleyre and F. Valois, "Virtual structure routing in ad hoc networks," in *IEEE International Conference on Communications (ICC)*, vol. 2, Seoul, Korea, 16-20 May 2005, pp. 3078–3082.
- [11] R. Krishnan and D. Starobinski, "Efficient clustering algorithms for self-organizing wireless sensor networks," *Elsevier Journal of Ad-Hoc Networks*, vol. 4, no. 1, pp. 36–59, January 2006.
- [12] R. Wattenhofer, "Algorithms for ad hoc and sensor networks," *Elsevier Computer Communications*, vol. 28, pp. 1498–1504, 2005.
- [13] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, 2002.
- [14] O. Younis and S. Fahmy, "HEED: A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Transactions on Mobile Computing*, vol. 3, no. 4, pp. 366–379, October-December 2004.
- [15] C. Dugas, "Configuring and managing a large-scale monitoring network solving real world challenges for ultra-low powered and long-range wireless mesh networks," *International Journal of Network Management*, vol. 15, pp. 269–282, 2005.
- [16] N. Patwari, J. N. Ash, S. Kyperountas, A. O. I. Hero, R. L. Moses, and N. S. Correal, "Locating the nodes - cooperative localization in wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 1, pp. 54–69, July 2005.
- [17] J. B. Saxe, "Embeddability of weighted graphs in k-space is strongly np-hard," in *17th Allerton Conference on Communications, Control, and Computing*, 1979.

- [18] S. Capkun, M. Hamdi, and J.-P. Hubaux, “GPS-free positioning in mobile ad-hoc networks,” in *Proceedings of the 34th Hawaii International Conference on System Sciences*, 2001.
- [19] A. Caruso, S. Chessa, S. De, and A. Urpi, “GPS free coordinate assignment and routing in wireless sensor networks,” in *Annual Joint Conference of the Computer and Communication Societies (INFOCOM)*. Barcelona, Spain: IEEE, 23-29 April 2006, pp. 150–160.
- [20] R. Fonseca, S. Ratnasamy, J. Zhao, C. T. Ee, D. Culler, S. Shenker, and I. Stoica, “Beacon vector routing: Scalable point-to-point routing in wireless sensor networks,” in *2nd Symposium on Networked Systems Design & Implementation (NSDI)*. Boston, MA, USA: ACM, 2-5 May 2005, pp. 329–342.
- [21] Q. Cao and T. Abdelzaher, “A scalable logical coordinates framework for routing in wireless sensor networks,” in *25th IEEE International Real-Time Systems Symposium (RTSS)*. Lisbon, Portugal: IEEE, 5-8 December 2004, pp. 349–358.
- [22] F. Benbadis, K. Obraczka, J. Cortes, and A. Brandwajn, “Exploring landmark placement strategies for self-localization in wireless sensor networks,” in *18th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. Athens, Greece: IEEE, 3-7 September 2007, pp. 1–5.
- [23] K. Liu and N. Abu-Ghazaleh, “Aligned virtual coordinates for greedy routing in WSNs,” in *International Conference on Mobile Adhoc and Sensor Systems (MASS)*. Vancouver, Canada: IEEE, October 9-12 2006, pp. 377–386.
- [24] B. Leong, B. Liskov, and R. Morris, “Greedy virtual coordinates for geographic routing,” in *IEEE International Conference on Network Protocols (ICNP)*, Beijing, China, October 2007, pp. 71–80.
- [25] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica, “Geographic routing without location information,” in *MobiCom*. San Diego, California, USA: ACM, September 14-19 2003.
- [26] N. B. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller, “Anchor-free distributed localization in sensor networks,” MIT Laboratory for Computer Science, Tech. Rep. 892, April 2003.
- [27] CC1101, *Low-Cost Low-Power Sub-1GHz RF Transceiver - Enhanced CC1100 (Rev. C)*, data sheet swrs061c [available online] ed., Texas Instruments, Inc., 27 May 2008.
- [28] T. Watteyne, A. Bachir, M. Dohler, D. Barthel, and I. Augé-Blum, “1-hopMAC: An energy-efficient mac protocol for avoiding 1-hop neighborhood knowledge,” in *International Workshop on Wireless Ad-hoc and Sensor Networks (IWVAN)*. New York, NY, USA: IEEE, June 2006, pp. 639–644.
- [29] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker, “Geographic routing made practical,” in *2nd Symposium on Networked Systems Design & Implementation (NSDI)*. Boston, MA, USA: ACM, 2-5 May 2005, pp. 217–230.