Robust Geo-Routing on Embeddings of Dynamic Wireless Networks

Dominique Tschopp, Suhas Diggavi, and Matthias Grossglauser

School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne (EPFL)
1015 Lausanne, Switzerland
Email: (first_name.last_name)@epfl.ch

Jörg Widmer
DoCoMo Euro-Labs
Landsberger Str. 312
80687 Munich, Germany
Email: (last_name)@docomolab-euro.com

Abstract— **Wireless routing based on an embedding of the connectivity graph is a very promising technique to overcome shortcomings of geographic routing and topology-based routing. This is of particular interest when either absolute coordinates for geographic routing are unavailable or when they poorly reflect the underlying connectivity in the network. We focus on dynamic networks induced by time-varying fading and mobility. This requires that the embedding is stable over time, whereas the focus of most existing embedding algorithms is on low distortion of single realizations of a graph. We develop a beacon-based distributed embedding algorithm that requires little control overhead, produces low distortion embeddings, and is stable. We also show that a low-dimensional embedding suffices, since at a sufficiently large scale, wireless connectivity graphs are dictated by geometry.**

The stability of the embedding allows us to combine geo-routing on the embedding with last encounter routing (LER) for node lookup, further reducing the control overhead. Our routing algorithm avoids dead ends through randomized greedy forwarding. We demonstrate through extensive simulations that our combined embedding and routing scheme outperforms existing algorithms.

I. INTRODUCTION

Mobile wireless networks comprise wireless devices with a limited transmission range, such as laptop computers, personal digital assistants (PDAs), cell phones, or embedded sensing and actuation devices. Such networks often rely on multi-hop communication, i.e., the forwarding of messages from a sender to a receiver outside the sender's radio range through intermediate nodes. One of the fundamental problems in ad hoc networking is thus the routing problem.

There are two approaches to the routing problem. The first approach relies uniquely on the topology of the network. Topology-based routing establishes an end-to-end path from a source to a destination through flooding or partial flooding of the network. Nodes either maintain a routing table that contains next hop and distance information for each destination [1], or the complete route can be stored in the header of all data packets [2].

The second approach exploits the geometry of the network. Nodes forward packets to neighbors geographically closer to the destination, assuming that the geographic proximity is representative of the network distance [3]–[5]. Geographic routing protocols require a location service [6], [7], which can be queried to obtain the location of other nodes.

Both approaches have advantages and disadvantages. The protocol overhead of topology-based routing stems from the flooding necessary for path establishment and maintenance. It is therefore very sensitive to mobility and uncertain channel environments, since even small changes in a node's neighborhood may lead to the failure of routes, which need to be reestablished. In contrast, geographic routing does not maintain state in the nodes, and forwarding decisions require only local knowledge, i.e., the geographic position of a node's neighbors. Topology changes that do not affect this local knowledge do not affect routing decisions, and therefore do not have to be advertised network-wide. However, geographic routing is inefficient when the network topology is not well captured by the geographic coordinates of nodes (e.g., due to a fading channel, obstacles, etc.). In such inhomogeneous networks, greedy routing towards the destination often reaches a local minimum, where no nodes with forward progress are known. Here, a recovery strategy is necessary, which requires either flooding or establishing state in nodes around the local minimum. This, together with the overhead introduced by the location service, may be more costly than the use of a topology-based routing protocol.

In this paper, we investigate how to bridge the two paradigms. More specifically, we are interested in building a virtual coordinate system that embeds the connectivity graph in a way that is coherent with the network topology. Nodes which are close in the topology should also be close in the embedding. Desirable properties of such a coordinate system are that it allows efficient greedy routing, is robust to mobility and channel uncertainty, and is cheap to maintain.

Embeddings of general graphs is a well studied topic, but few algorithms specifically for wireless graphs exist. Furthermore, ad hoc networks require an efficient *distributed* implementation of the embedding algorithm. Our main objective is to create embeddings which provide efficient routes when combined with greedy routing. This also reflects on the choice of the metric with which the wireless graph is embedded. The intuition which drives the techniques used for our embedding algorithm is the following. We believe that geometry plays an important role for long paths, whereas short paths are more subject to local perturbations.

The paper is structured as follows. Related work is discussed

in Section II, and some background on graph embeddings is given in Section III. In Section IV, we develop a random beacon-based embedding with the following features: (i) approximates well the single snapshot graph distances, (ii) gives a stable coordinate system when embedding dynamic graph topologies, and (iii) has low overhead in terms of network-wide control traffic. This embedding algorithm is then combined with a novel randomized greedy routing algorithm in Section V. The idea behind the randomized routing strategy is also independently applicable to other scenarios with uncertainties in the topology or location. We give extensive simulation results to validate the properties of our algorithms in Section VI.

II. RELATED WORK

Graph embeddings are an active area of research with many different applications [8]. Some of the concepts of this paper, for example, are based on embedding algorithms used for the analysis of molecular similarities [9]. For networking, embeddings have been studied mainly in the context of mapping network distances in the Internet to Euclidian space [10], [11], and relatively few embeddings exist that are specifically designed for routing in wireless ad hoc networks. The two schemes most closely related to the algorithms proposed in this paper are the pioneering work presented in [12] and the beacon vector routing (BVR) introduced in [13].

In [12], a routing scheme (NoGeo) with a distributed relaxation algorithm is presented. It iteratively builds a virtual coordinate system. The algorithm assumes that a number of so-called perimeter nodes (located on the border of the network) know their real coordinates in advance. Starting from random virtual coordinates, at every iteration, non-perimeter nodes update their coordinates by averaging the coordinates of their neighbors, while the coordinates of perimeter nodes remain fixed. The scheme also includes a flooding-based mechanism to determine identity and/or coordinates of the perimeter nodes if they are not known in advance. The protocol performs well in terms of success rate of greedy routing, but the overhead to maintain perimeter nodes is very high when the topology is dynamic. The overhead also grows superlinearly with the size of the network due to the increased length of the perimeter. The algorithm critically depends on correct perimeter node information and at lower overhead may lead to a completely distorted coordinate system.

BVR [13] is intended for routing in sensor networks. Here, the hop distances to beacon nodes directly form the virtual coordinates of a node, and the dimensionality of the coordinate space corresponds to the number of beacons. The set of beacons is randomly chosen and does not change unless a beacon fails. As in [12], greedy forwarding over the virtual coordinates is used as routing scheme. A distance metric is defined that takes into account that greedy routing in the direction of a beacon is more likely to lead to the destination than routing away from a beacon (explained in more detail in the simulation section). In case of a dead-end, a small scope flooding is initiated to find a node that again provides greedy

progress. The scheme is very sensitive to a bad inital choice of beacons. It also does not cope well with mobility of beacons or an uncertain channel environment, both of which lead to large shifts in the virtual coordinates. This not only results in unstable routes, but also incurs a high cost for updating a location service with the changing positions of the nodes.

There are a number of further embedding algorithms for routing in sensor networks. In [14], a ringed tree graph is built for data-centric information processing with coordinates that are similar to polar coordinates. Both MAP [15] and Glider [16] build a virtual coordinate system based on a tiling of the network area. The former builds a backbone structure adapted to the shape of the network topology and connects sensor nodes to the nearest backbone node through shortest paths. The latter uses Delaunay triangulation to form Voronoi cells and routes toward a so-called landmark node of the adjacent Voronoi cell that lies in the right direction. Finally, some schemes use multidimensional scaling techniques to build a coordinate system from connectivity information (e.g., [17]). All of these systems have in common that they are not designed to cope well with mobility or varying channel conditions. While the former may be less important for sensor systems (but is very important for example for vehicular ad hoc communication), the latter is an inherent property of all wireless networks.

III. LOW-DIMENSIONAL EMBEDDINGS

In this section, we review recent results in embedding theory that motivate the beacon-based embedding algorithm presented in the next section, and provide some intuition for its design. Over the past decade, significant progress has been made in both algorithms and bounds for embeddings of (finite) metric spaces (see for example [8] and references therein). Given the distances $D(\cdot, \cdot)$ between n points in a metric space (X, D), the goal of the embedding is to find a mapping $x^{'} = f(x), x \in X, x^{'} \in X^{'}$ from X to another space $X^{'}$ such that for the metric $D^{'}(\cdot, \cdot)$ in $X^{'}$, for all points $x, y \in X$, the distances $D^{'}(f(x), f(y))$ do not distort $D(x, y)$ very much. More precisely, the embedding $f(\cdot)$ is said to have distortion at most c if there is a $r \in (0, \infty)$ such that for all $x, y \in X$, $rD(x, y) \leq D^{'}(f(x), f(y)) \leq crD(x, y)$. For $c = 1$, the embedding is called non-contracting. Also, typically the target space $X^{'}$ is a Euclidean space with $D^{'}$ being the l_2 norm. In such cases we can talk about the dimension of the embedding to be the dimension of $X^{'}$.

In the aforementioned framework, many graphs[1] do not admit low-distortion and low-dimensional embeddings simultaneously. There has been a significant amount of effort in classifying the distortion and dimension of specific classes of graphs (see Table 8.5.1 in [8]). This leads to the notion that in general graphs are not "embeddable", *i.e.,* do not admit low-distortion embeddings in low-dimensional spaces. More recently, an alternate question was posed in [18], which introduced the notion of *slack* in embeddings. Slack allows a

---

[1]We interchangeably use the term graph for a finite metric space.

small fraction of the distances to be arbitrarily distorted, while the others are guaranteed to have a much smaller distortion. In particular, [19] showed that every finite metric space can be embedded into a l_p space with constant dimension $O(\log^2(\frac{1}{\epsilon}))$ with constant distortion $O(\log(\frac{1}{\epsilon}))$, if a fraction ϵ of the distances in the original space can be arbitrarily distorted. However, the distortion may not be uniform across nodes. Therefore we need a stronger notion of slack called *uniform slack* which means that for every point $u \in X$, at most fraction ϵ of its pairwise distances can be arbitrarily distorted. Embeddings with uniform slack is also explored in [19].

One of the main techniques in embeddings with slack is the use of a constant number of beacons for the embedding. Such an embedding is based on triangulation, *i.e.,* reconstructing the distance between two non-beacon points from their known distances to a set of beacons. Clearly, for points that are close to each other, this can cause arbitrarily large distortion. Hence, these pairs of points are counted towards the uniform ϵ slack.

The notion of slack is inherently useful for embeddings of wireless network graphs. This is because a small number of edges suffice to transform a graph admitting a low-dimensional embedding into a graph that does not. The randomness in node locations as well as mobility and channel uncertainty (fading) can easily perturb the original geometry enough to make the graph difficult to embed completely. However, our intuition is that for wireless graphs, even though local geometry is easily perturbed through these sources of randomness, at large scales this would matter much less. Therefore, slack eliminates some of the local behavior, and allows to embed the larger-scale distances into a low-dimensional space with low-distortion. We also use this principle of slack in Section V, where we develop routing algorithms.

We also note that graphs that do not possess low-dimensional embeddings can arise even without channel uncertainty. Specifically, it is possible to construct unit-disk graphs (UDG), where an edge between two nodes u and v exists if and only if $||X_u - X_v|| < 1$, for which no low-distortion, low-dimensional embeddings exist [20]. However, these require specific node constellations that occur only with very low probability in a random realization of node locations. Hence, these constructions are mostly of theoretical interest.

A. Stable Embedding of Connectivity Graph

We have discussed above the classical embedding problem of a metric space (X, D) into another (usually normed) space (X', D') through an embedding function $f : X \rightarrow X'$, and the associated metrics for the quality of the embedding (stretch, slack).[2] We now introduce a novel aspect of the embedding problem discussed in this paper: maintaining a stable embeddings of a dynamic graph.

Indeed, the connectivity graph of a mobile wireless network changes over time because of node mobility and channel uncertainty. We can view this as a dynamic metric space

---

[2]We discuss other metrics from the field of *multidimensional scaling* in [21].

(X, D_t), for which we would like to maintain an embedding $f_t(.)$. We define the embedded distance between x_1 and x_2 at time t as $d'_t = D'(f_t(x_1), f_t(x_2))$.

What is a good dynamic embedding? For obvious reasons, we would like the dynamic embedding $f_t(.)$ to be a faithful representation of (X, D_t) for every time t. However, this is not sufficient for our purposes, because it does not say anything about the evolution of the embedding over time. For example, even if the metric space (X, D_t) were fixed over time, the embedded coordinates might fluctuate, provided the distances $d'_t(., .)$ remain stable. This is undesirable, for the following reason.

In our setting, although the graph changes over time, it tends to change slowly. For example, two nodes that are far apart at time t are unlikely to be very close a short time after t, and vice versa. This is a result of physical constraints on node mobility processes (nodes cannot jump from one place to another), and the fact that channels between nodes strongly depend on geography, as explained above.

Therefore, the distances D_t change slowly over time, with the largest relative changes concentrated on short distances. The stable embedding problem amounts to maintaining a dynamic embedding such that (a) the instantaneous distances $d'_t(., .)$ are close to the real distances $D_t(., .)$ for every t, and (b) the coordinates $f_t(.)$ of the embedded space changes as slowly as possible.

A stable embedding in our context is important for the following reasons. A geo-routing algorithm has to be paired with a location service in order to be able to deliver messages to particular nodes (or information items), rather than to particular locations. A location service is essentially a distributed database that maintains the location of every node in the network. The database has to be updated when the location of a node changes. Suppose we have a node x that does not move, but whose coordinates in the embedding $f(x)$ change over time; then updates would have to be generated continually for this node, resulting in overhead. A stable embedding minimizes this overhead.

A slightly different approach eliminates the need for location updates by merging the location service into the routing protocol. In this approach, a message starts out with an imprecise estimate of the destination's location. It then refines this estimate as it travels through the network. It has been shown that it is sufficient (depending on the mobility process) that each node remembers when and at what location it was last a neighbor of every other node. This approach, called Last Encounter Routing (LER), amounts to the message traveling towards past locations of the destination. If LER operates on embedded coordinates, then a stable embedding ensures that these past locations are close to the current location of the destination. Even though there is no need for location updates in LER, an unstable embedding would manifest itself through increased route cost, as the message would frequently move "in the wrong direction".

This illustrates that stable embeddings are important to minimize overhead in the context of geo-routing. Therefore, in

this paper, our goal is to develop a distributed algorithm that computes faithful and stable embeddings for slowly changing graphs.

In summary, the results on graph embeddings provide important insights for the design of the algorithms in IV and V. In the presence of mobility, designing a beacon management method that produces stable embeddings is challenging. Slack is an important concept for embeddings of wireless graphs since there are low probability events that can significantly deteriorate embedding performance. In Section IV we explore methods that do local operations to account for such inaccuracies. There is always some degree of distortion in embeddings and routing protocols need to take this into account. To this end, we propose a randomized algorithm scheme in Section V.

IV. EMBEDDING ALGORITHM

In this section, we describe our distributed probabilistic beaconing (PB) algorithm. We first provide some intuition on the design of the algorithm, motivated by the discussion in the previous section. We then formally define the algorithm and explain its operation through an example.

A. Embedding Heuristic

The basic idea of our algorithm is to use random beacons as anchors of an embedding. The algorithm is specifically designed to maintain a stable embedding when the graph changes over time, and to combine global beaconing with local correction operations to restore local geometry as far as possible.

We now describe the heuristic for a node i to compute its current embedded position x_i for an embedding in M dimensions. We assume that a node i has information from a set $\mathcal{B}$ of beacons for which it knows both its graph distance h_{B_l}, $B_l \in \mathcal{B}$ and x_{B_l}, the embedded coordinates of the beacon $B_l \in \mathcal{B}$. From this, the node attempts to find its embedded coordinates x_i using the following criterion.

$$\min_{x_i} \sum_{B_l \in \mathcal{B}} [h_{B_l} - ||x_i - x_{B_l}||]^2 \overset{def}{=} \min_{x_i} h(x_i). \qquad (1)$$

We use a heuristic to solve this optimization problem because $h(\cdot)$ is a non-convex function. Suppose node i knows its graph distance to a set of beacon nodes $\mathcal{B} = \{B_1, \ldots, B_b\}$. An iterative heuristic to solve (1) will update the position x_i of node i by moving it towards or away from the beacons. This is reminiscent of stochastic proximity embedding (SPE) [22].

If the wireless connectivity graph G were perfectly embeddable in two dimensions, and given the positions and the shortest path distances to at least three beacons, the positions of all other nodes would be uniquely determined. In reality, we expect G to be low dimensional only within some slack ϵ. Therefore, G requires an embedding dimension of at least $M = 2$, with a small incremental benefit for higher-dimensional embeddings ($M > 2$). In the PB algorithm, after the flooding of the third beacon, this manifests itself by the nodes clustering close to a two-dimensional hyperplane in
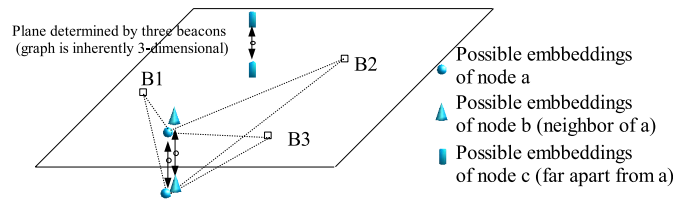

Fig. 1. Nodes a and b are neighbors, while node c is far away from both. The two optimal positions of node a and b are close. In the worst case, a and b are placed on opposite sides of the plane and the relative error is approximately 1 over the distance between these optimal positions. c being placed further apart, the relative error in the distance is small. B1,B2 and B3 are beacons.

the M-dimensional space. The relative error in the distances between non-beacon nodes is bounded by the variance in higher dimensions for nearby nodes, and becomes negligible for nodes far apart. An illustrative example where G is inherently 2-dimensional with low variance in the third dimension is shown in Figure 1.

B. Dealing with Dynamic Graphs

Under mobility and channel uncertainty, large scale distances remain relatively unaffected over short time scales. We exploit this slow evolution by updating the embedding in a lazy manner, but giving up on short distances. Specifically, we propose to use a sliding window mechanism to update the embedding in the face of graph dynamics. Every time the distance to a new beacon is learned, the oldest distance is thrown away. By only changing one of the beacons at a time, the coordinate system cannot change drastically as the other beacons remain fixed. Always choosing new random beacons ensures that we are not dependent on the initial choice of beacons. A new random choice at every iteration guarantees a good performance on average.

We include an additional mechanism to stabilize the embedding. To make the embedding locally consistent, nodes estimate their distances to the beacons as the average of their observed distance and the observed distances of their one-hop neighbors. The underlying idea is that when a node moves into a new neighborhood, its distance estimate should be close to the distance estimates of its new neighbors. Local coherence is especially important for geographic routing, where all forwarding decisions are local.

C. Formal Description of PB Algorithm

At every time step k, a new beacon node is randomly selected. This beacon floods the network with a control message, through which each node learns its shortest-path distance in G to the new beacon.

Our heuristic updates node i's embedded position x_i, by iteratively minimizing the criterion given in (1) using a gradient-descent technique. Nodes know their Euclidean distance and graph distance to beacons and consequently new beacons can be chosen with a higher probability if they are well embedded with respect to already existing beacons. Adding new beacons will push nodes out of local minima. To increase the probability that neighbors have similar coordinates (e.g.,to

move to the same side of the plane in Figure 1), the input hop-distance to the algorithm is the average of a node's and its neighbors' hop-distances.

At every time k, a randomly selected beacon B_k floods the network with its virtual position $x_{B_k}^{(k)}$, where the superscript indicates the time-index of the coordinate. All other nodes in the network obtain their hop distance, or *proximity*, $P_i(k)$ to B_k in this way. We initialize $x_{B_0}^{(0)}$ with a random M-dimensional vector.

Nodes have a buffer in which they store their proximities (hop-distances) to the b last beacons as well as the virtual positions of these beacons. Let us call $\mathcal{B}(k)$ the set of the b last beacons at time k, and $P_i(k-l)$ the proximity of node i and beacon B_{k-l} at time $k-l$. Let $E_{ij} = ||x_i - x_j||$ denote the Euclidean distance between nodes i and j with positions x_i and x_j (in the virtual/embedded space) and $\mathcal{N}_i$ the one-hop neighbors of i. The probabilistic beaconing algorithm is shown in Algorithm 1. Nodes temporarily store a vector containing the average of their distances to the beacons and the distance of their one-hop neighbors to the beacons. Then, nodes iteratively project their position on a hypersphere around every beacon of radius equivalent to the previously estimated distance $\hat{h}$ to that beacon. The input to the next iteration is the average of the projections. Figure 2 illustrates two iterations of the gradient-descent algorithm in a two dimensional space with three beacons.

Fig. 2. Two iterations of the PB algorithm in 2 dimensions. The node projects itself on a hypersphere of radius h_u around every beacon B_u. The input to the next iteration is the average of these projections. This implements a gradient-descent minimization of $h(\cdot)$ defined in (1).

Algorithm 1 Probabilistic beaconing

---

1 At time k obtain distance $P_j(k)$ of vertex v_j to beacon B_k

2. Adjust distances to beacons, averaged over neighborhood
   For $u = 0$ to $b-1$
   $\quad$ Set $\hat{h}_u(k) = \frac{1}{|\mathcal{N}_i|+1} \left(\sum_{j\in\mathcal{N}_i} P_j(k-u) + P_i(k-u) \right)$
   end

3. Local optimization
   Starting point is center of mass of node i + neighbors
   $x := \frac{1}{|\mathcal{N}_i|+1} \left(\sum_{j\in\mathcal{N}_i} x_j^{(k)} + x_i^{(k)} \right)$
   Repeat W times
   $\quad x := \frac{1}{b} \sum_{u=0}^{b-1} x + \left(\hat{h}_u(k) - ||x - x_{B_{k-u}}^{(k-u)}|| \right) \frac{x - x_{B_{k-u}}^{(k-u)}}{||x - x_{B_{k-u}}^{(k-u)}||}$
   end

4. Update position
   Set $x_i^{(k+1)} := x$

V. ROUTING ALGORITHM

As we have explained, geometry plays a role for large distances, such that a small number of beacons are sufficient to embed at a low distortion and roughly place nodes at the correct location. To accurately embed short distances, practically every node would have to be a beacon and the dimension of the embedding would be much higher, which is not a desirable property for an embedding algorithm. We propose two local mechanisms to cope with these inconsistencies in the coordinate system.
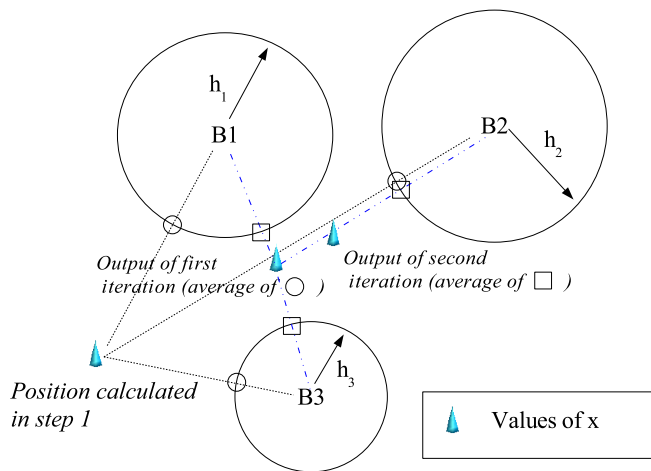
The first mechanism is based on the principle of slack introduced in Section III, which showed that the largest distortion of the distances occured in the local neighborhood. To overcome these errors, the idea is that every node can build a local routing table that is sufficiently large to overcome this local inaccuracy of the coordinate system. Therefore, the size of the slack is related to the size of such a local routing table.

Alternatively one can use Biased Random Walk (BRWalk) explained below. BRWalk is designed for coordinate system which are slightly inaccurate, such as the one obtained with PB or noisy samples of real coordinates in a dense network (without large voids). It trades off path length for robustness by not trusting the coordinate system completely. In BRWalk, the next hop is chosen randomly among all neighbors. By introducing randomness into the routing decisions, we tolerate some "wrong" forwarding decisions which on one hand increase the path length but on the other hand allow to transparently avoid getting stuck in a dead-end. Nodes which are geographically closer to the destination than the current node have a higher probability of being selected as next hop, but going "backward" and loops are not excluded. In order to reduce the probability of visiting the same node several times, one can store a constant size list of the last visited nodes and reduce the probability of returning to these nodes.

Assume a source s has a packet for a destination t. For every node $j \in \mathcal{N}_s$, s computes the difference between its Euclidean distance to the destination and the Euclidean distance of node j to the destination $\Delta_j = E_{st} - E_{jt}$. Then, the probability of choosing node j as a next hop is given as $p_j = \frac{f(\Delta_j)}{\sum_{k\in\mathcal{N}_s} f(\Delta_k)}$. Additionally, if the packet can hold the identifier of the last n hops in a variable $path$, one can modify the probability of visiting a node which occurs several times in the path *i.e.*,

$$p_j = \frac{f(\Delta_j, path)}{\sum_{k\in\mathcal{N}_s} f(\Delta_k, path)} \qquad (2)$$

One possibility is to set $f(\Delta_j, path) = \frac{e^{\alpha\Delta}}{2^m}$ where α is a parameter determining the "greediness" of the routing and m is the number of times node j appears in $path$. Note that when $\alpha \to 0$, the routing algorithm simply performs a random walk and that when α becomes large enough, BRWalk routing is equivalent to greedy routing (in this case the $path$ is not taken into account). The next hop is a sample drawn according to the distribution given in (2). Note that in this routing algorithm, packets need to have a time to live (TTL) field, as in the worst case, with low probability they might never reach the destination. We consider that a small percentage of lost packets is acceptable if it allows us to considerably reduce the overhead.

VI. Simulation Results

In this section we evaluate our algorithms through a series of simulations using a custom discrete time simulator. In every round, nodes first move, then update their positions, and then communicate. A node can communicate with any other node in the network during such a round, potentially over multiple hops. We assume for simplicity that there is no packet loss at the MAC layer.

A. Experiment Design

Our embedding algorithm is designed to cope with long term fading rather than with short term fading. Consequently, we consider that nodes can move but that the channels, which are determined by the environment, do not change over time. The network model we use therefore consists of an $S \times S$ grid of locations. Every location can be occupied by none, one or several nodes. The channel existing between the locations is drawn *a priori*. Hence, if a node i occupies a location l_1 and another node j occupies a location l_2, these nodes will be directly connected through an edge (i, j) if a link exists between the two locations l_1 and l_2. In our simulation we consider that every location picks λ other locations according to an exponential distribution with mean r for the distance and at angles chosen uniformly at random around itself to connect to.[3] Note that this is not a unit disk graph model (UDG)[4], and consequently routing algorithms tailored for this particular class of graph are not applicable (*e.g.,* planarization in [3]–[5]). To simulate node mobility, we use the random walk (RW) model and the random waypoint (RWP) model. When a node moves, its coordinates are rounded to the closest grid location. We compare our approach with BVR [13] and with the averaging (NoGeo) approach proposed in [12]. For NoGeo, unless stated otherwise, we consider that the perimeter nodes as well as their positions are known. We made that choice as applying the "perimeter node" criteria described in [12] led to a very large amount of falsely detected perimeter nodes

and in turn to very poor performance with the random channel model, especially in mobile scenarios. By default, we consider 20 perimeter nodes. Unless stated otherwise, we consider a network of size $S = 30$ with $N = 1500$ nodes with $\lambda = 10$ and an expected communication range of $r = 1.5$. As a general rule, we allow an overhead of 10 messages per node to build the embeddings per round (10 averaging steps for NoGeo, 10 beacons can flood for BVR and PB). In BVR the dimension of the embedding is equivalent to the number of beacons. To make the comparison with PB meaningful, we use the same dimension of embedding for PB and BVR and 2 for NoGeo. We also use $b = 20$ for PB. The dimension of embedding M can be considered low if it is constant and $M \ll N$, where N is the number of nodes. The dimension M set to 20 by default for PB and BVR.

B. Performance Metrics

We evaluate embedding algorithms according to the following criteria:

1) Distortion: Given two nodes i and j with virtual coordinates x_i and x_j, we define the multiplicative distortion as $\frac{||x_i - x_j||}{r_{ij}}$ where r_{ij} denotes the shortest path distance in the connectivity graph between i and j.

2) Greedy Routing Success Rate (GSR): The fraction of packets that reach their destination by making only local forwarding decisions based on the coordinates of the nodes in the neighborhood and the position of the destination. For PB and NoGeo, we use classic greedy routing which forwards a packet to the neighbor closest in Euclidean distance to the destination. For BVR, we use both greedy routing and the routing algorithm proposed in [13], here called "BVR greedy" and "BVR", respectively. For the latter, the distance between two nodes p and d is given by $A\delta^+ + \delta^-$, where $\delta^+(p,d) = \sum max(p_i - d_i, 0)$ and $\delta^-(p, d) = \sum max(d_i - p_i, 0)$. Index i corresponds to the i^{st} coordinate. As in [13], we set weight $A = 10$ and take all beacons into account. The neighbor that minimizes this distance function is chosen as a next hop.

3) Path Stretch: The path stretch is the ratio between the actual number of hops a data packet traveled from a source to the destination and the shortest path distance in hops between these two nodes. We only consider successful communication.

4) Communication Overhead: We consider as overhead all packets that are not data packets. This includes the packets flooded by beacon nodes as well as the packets used to build local routing tables and the packets used in ring search.

5) Virtual speed: This metric captures how fast nodes move in the virtual coordinate space, *i.e.,* the average Euclidean distance between the virtual coordinates of nodes from one round to the next.

C. Static Networks

In this section, we investigate the performance of PB in static networks both in terms of embedding quality and routing efficiency.

---

[3]Channels are considered to be bidirectional, so that a location which is "chosen" by another node can have a degree higher than λ, while a node that randomly picks several times the same location might have a degree lower than λ. We bound connectivity to λ locations for simplicity, and verified that the results are equivalent to an exponential distribution over all points.

[4]In a UDG model, nodes i and j with positions x_i and x_j respectively are connected if and only if $||x_i - x_j|| \leq r$, for a fixed communication radius r.

1) Quality of Embedding: In Figure 3(a), we show the empirical cumulative distribution function (ecdf) of the multiplicative distortion in a fixed network size. It can be seen that it is low with PB is low. This indicates that PB can efficiently capture the inherently low dimensional structure of wireless connectivity graphs. Further, a small number of beacons suffice to this end. One can also point out the fact that the slope of the cumulative distortion curve with PB is almost vertical which indicates a lower variance. It is interesting to note that the distortion of BVR is high, since nodes are spread out in all dimensions. Due to the randomness of the channel, there can also be highly distorted distances with the real coordinate system (*e.g.,*if nodes located in neighboring locations are not connected directly). With NoGeo, the averaging procedure can place nodes arbitrarily close or far apart so that a fraction of distances are highly distorted. As shown in Fig. 3(b), the mean multiplicative distortion does not appear to grow considerably with the size of the network. In addition, it stays very close to 1 with PB, which tends to indicate that even the additive distortion is small. A direct consequence will be that geographic routing performs well on top of a virtual coordinate system built with PB, independently of the size of the network.

(a) Multiplicative distortion   (b) Mean multiplicative distortion versus network size
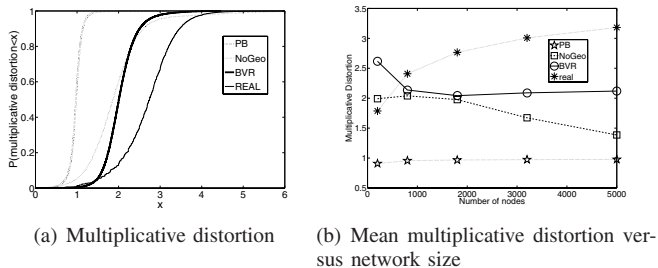
Fig. 3. Cumulative distribution function of multiplicative distortion for 2000 nodes and mean distortion as a function of number of nodes. When we increase the number of nodes, we maintain a constant density of 2 nodes per grid location.

2) Quality of Routing: We will first investigate the performance of greedy routing on top of virtual coordinate systems in a static setting. In particular, we will focus on the performance of the algorithms in networks of increasing size, networks of increasing node density and in inhomogeneous network topologies. We will also investigate local optimization mechanisms to improve the quality of routing.

a) Scalability, Density, and Inhomogeneity: Increasing the size of the network while maintaining a constant density of 2 nodes per grid location has the effect of introducing larger distances in the topology. The overhead per node allowed to build the embedding is kept constant. A consequence is that in PB and BVR, the beacons are spread out. We show in Figure 4(a) that this affects the GSR of PB only marginally. On the other hand, a clear effect can be seen with NoGeo as the node positions depend on the position of perimeter nodes and the averaging takes more time to reach the center of the network. A direct consequence of the way the BVR embedding is built is that is is easier to route toward beacons. Indeed, one can observe that the GSR decreases remarkably with this approach

when we increase the network size. A similar phenomenon can be observed when the network area is reduced and the number of nodes is kept constant. Figure 4(b) shows that NoGeo and also BVR are severely affected by changes in the network diameter, while PB is relatively unaffected. Increasing the network size also creates voids in the topology as not all grid locations are occupied anymore, which reduces the GSR of real coordinates. This effect can also be seen in topologies with obstacles (see Figure 4(c)). Note that the GSRs of NoGeo and BVR are fairly low since we limit all approaches to the same overhead to build the embedding and the convergence speed of PB is higher. Our simulations have shown that in order to reach the same GSR as PB, NoGeo and BVR need an overhead of up to 100 messages per node in this setting.

b) Local Optimizations: As explained in III, small distances are hard to embed. As explained in Section V, allowing nodes to build local routing table of growing scope can increase the GSR and reduce the path stretch remarkably as shown in Fig. 5(b) and 5(a), but the overhead to build the routing tables increases likewise.

BRWalk allows to trade path length for overhead. In Figure 5, we show that for a sufficiently high value of the parameter α (see Section V), the stretch can be as low as 2 while the GSR reaches 100%, which is similar to the one obtained with local routing tables at no cost in terms of overhead. This shows that when a good next hop is available that is much closer to the destination than the current node, this node should be chosen. In other words, if we make a big progress in Euclidean space, we should trust the embedding. On the other hand, when no such node exists, a next hop should be chosen randomly. This result also suggests that there are small errors in the embedding. The ordering with respect to a target node of nodes embedded nearby can be perturbed. In turn, this phenomenon leads to wrong forwarding decision. Introducing randomness in the forwarding decisions appears to be an efficient way to mitigate the effects of such disorderings.

D. Mobile Networks

Here, we study the behavior of embedding algorithms when the wireless connectivity graph is dynamic because of node mobility.

1) Quality of Embedding: In mobile networks, we measure the quality of embedding by looking at the average virtual speed of nodes in the virtual coordinate system. Only a certain amount of "fresh" distance information is injected per round. In this case, the embedding is partly built based on outdated distance information. The virtual speed with PB is of the same order as the real speed of the nodes. The sliding window mechanism mitigates the effect of injecting new distances while the averaging mechanism mitigates the local incoherence in the coordinate system. Again, we assume that for NoGeo perimeter nodes are given, since the perimeter node detection procedure in [12] (a node is a perimeter node if it is further away from a beacon node than all of its two hop neighbors) leads to a large number of false positives and ultimately to the collapse of the coordinate system. In Figure 6(a), we show the

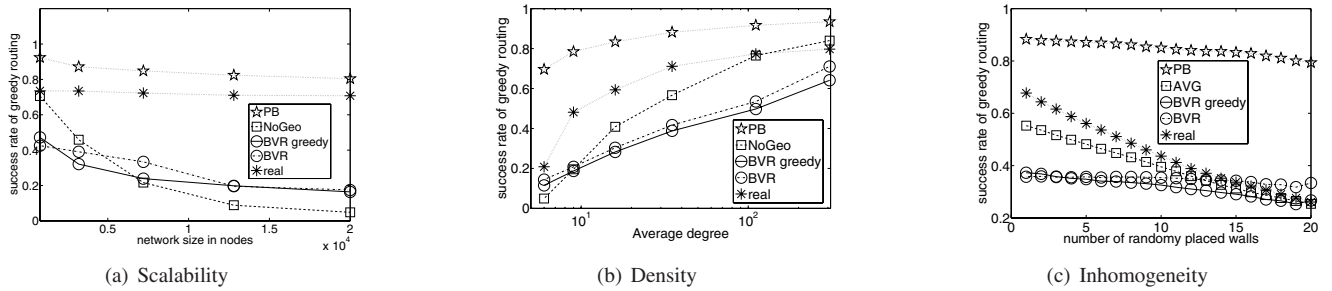(a) Scalability   (b) Density   (c) Inhomogeneity

Fig. 4. GSR as a function of network size, node density and number of obstacles (randomly placed straight walls of length 6). The overhead allowed to build the embedding is set to 20 messages per node.
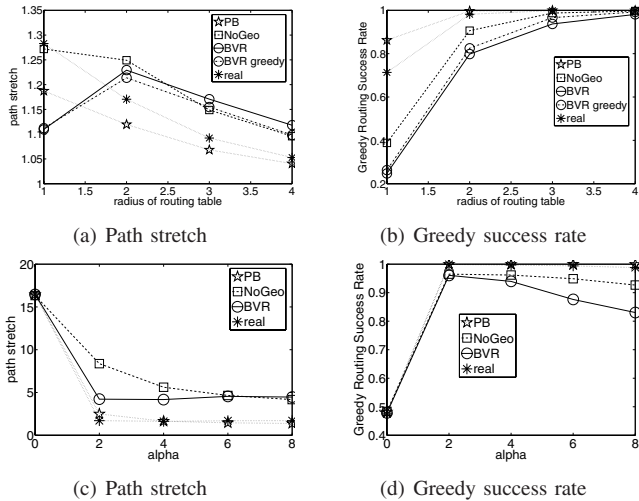

(a) Path stretch   (b) Greedy success rate

(c) Path stretch   (d) Greedy success rate

Fig. 5. GSR and path stretch for BRWalk routing and forwarding with local routing tables. α captures the randomness of the forwarding in BRWalk. A greater value for α means less randomness. In this experiment, packets have a memory of 20 hops, as explained in Section V.

average virtual speed of nodes when the nodes move according to the RW and RWP model with a speed of 1.
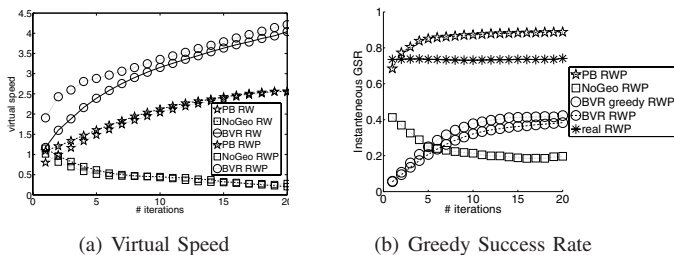

(a) Virtual Speed   (b) Greedy Success Rate

Fig. 6. Virtual Speed and GSR under mobility as a function of the allowed number of iterations to update the coordinate system between every move

2) Quality of Routing: Second, we analyze the quality of routing in a dynamic setting.

a) Instantaneous GSR: We investigate the performance of PB when we limit the number of iterations to update the algorithm in every round. Figure 6(b) shows that as few as 5 iterations are sufficient to obtain a high GSR. For the sake of clarity we only show the results for the RWP as the RW results are very close. It is remarkable that even under mobility a

higher GSR than with real coordinates can be obtained. There is an interesting trade-off between stability and GSR. The more fresh information is added to the embedding, the more it moves but the better the GSR. In a static environment, BVR is more stable since the beacon nodes never change, but the GSR of PB can still be up to two to three times higher than that of BVR, depending on the number of beacons. In a mobile environment, however, PB is both more stable and more efficient in terms of GSR. Note that the GSR of NoGeo decreases because new perimeter nodes are selected continuously, and the coordinate averaging does not manage to propagate through the network sufficiently fast to keep up with the new information.

b) Reliable routing: We now investigate reliable routing under mobility. When a packet is stuck with greedy forwarding, a ring search is started until a node closer to the destination is found for NoGeo and PB. For BVR, we use the recovery strategy proposed in [13] which routes a packet toward the beacon closest to the destination. If along the way, the packet reaches a node which is closer to the destination than the dead-end, greedy mode is resumed. Otherwise, a ring search is started from the beacon. In Fig. 7, we are interested in how the communication overhead to achieve reliable communication scales with the size of the network. We allow only a fixed overhead per node to update the embedding in every round, and consequently the embeddings will be based on outdated distance information. In Fig. 7(a), it can be seen that reliable communications are less costly in terms of overhead with PB than with the other approaches. Note that the overhead with real coordinates remains low as there is no additional cost to update the real coordinate system. The gap between the different schemes grows as the size of the network increases. Fig. 7(b) and Fig. 7(c) give us some insight. In the former, it can be seen that the number of dead-ends per packet sent remains very low with PB. In the latter, it is shown that when such a dead-end occurs recovery is cheap. Indeed, on average a node only needs to search its two hop neighborhood to find a suitable next hop. With the other approaches, the number of dead-ends per packet increases with network size as well as the search radius necessary to find a next hop. It is also worthwhile to note that the path stretch with PB remains more or less constant. These are indicators that the quality of the embedding and its stability do only scale well with PB.

(a) Overhead per packet

(b) Dead-ends per packet

(c) Flooding radius per dead-end
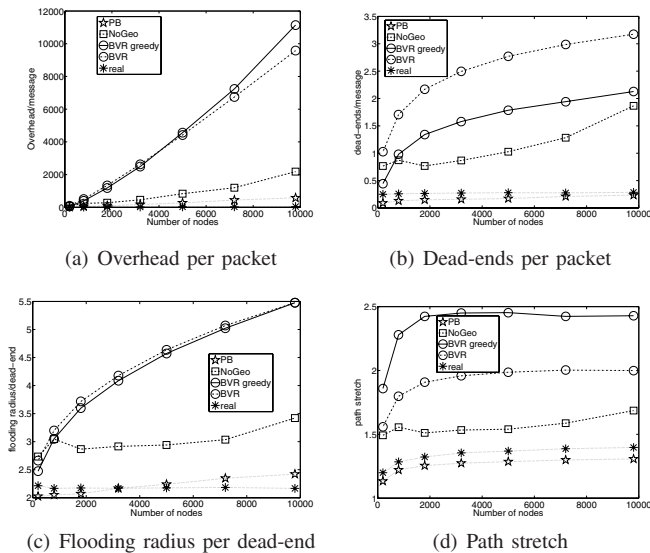
(d) Path stretch

Fig. 7. Reliable routing under mobility. Nodes move according to the RWP model with speed 1 and pause time 0. The are on average 2 nodes per grid location and we maintain this density fixed while we increase the number of nodes. Nodes are allowed an overhead of 5 messages to update the embedding before they move again (5 iterations). The dimension of embedding is 20 for BVR and PB and 2 for NoGeo

c) Location service: Overall, our simulations show that the embeddings built with PB are sufficiently stable to be used with Last Encounter Routing (LER), resulting in a complete solution for routing in dynamic wireless ad hoc networks with low overhead. As expected, PB is particularly efficient compared to real coordinates in inhomogeneous topologies. Indeed, our experiments have shown that in settings with obstacles the overhead to route a packet with a PB embedding and LER (including the overhead to build the embedding) remains lower than the overhead necessary to route a packet with LER on top of real coordinates. This remains true when we increase the network size. We refer the reader to [21] for a more in depth explanation of LER on top of PB. Note that when the topology is fixed, all embeddings oscillate very little and are consequently also adapted for classical location services (LS). Indeed, location services updates are in general triggered when a node has moved a certain distance.

VII. CONCLUSIONS

In this work we present a novel embedding algorithm for greedy routing on virtual coordinates that is robust to network dynamics and random channel conditions. Its key feature is the intricate combination of local information (averaging) and global information (probabilistic beaconing) in an iterative, distributed manner. We give some intuition why these connectivity graphs are inherently well embeddable even in the face of mobility. The proposed algorithm significantly outperforms existing embeddings in terms of success rate of greedy routing, convergence and distortion of the embedding, and overhead in static as well as dynamic environments. We further show that some degree of randomness in the routing decisions alleviates the effects of local inaccuracies in the positions of the nodes.

REFERENCES

[1] C. E. Perkins and E. M. Royer, "Ad hoc on-demand distance vector routing," in *Proc. of 2nd IEEE Workshop on Mobile Computing Systems and Applications*, Feb. 1999.

[2] D. B. Johnson, D. A. Maltz, and J. Broch, *Ad Hoc Networking*. Addison-Wesley, 2001, ch. DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks, pp. 139–172.

[3] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, "Routing with guaranteed delivery in ad hoc wireless networks," in *Proc. of 3rd ACM DIALM*, Seattle, US, 1999.

[4] B. Karp and H. T. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in *Proc. ACM Mobicom*, Aug. 2000.

[5] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger, "Geometric ad-hoc routing: Of theory and practice," in *Proc. ACM PODC*, July 2003.

[6] S. M. Das, H. Pucha, and Y. C. Hu, "Performance comparison of scalable location services for geographic ad hoc routing," in *Proc. IEEE Infocom*, Mar. 2005.

[7] M. Grossglauser and M. Vetterli, "Locating Mobile Nodes with EASE: Learning Efficient Routes from Encounter Histories Alone," *IEEE/ACM Trans. on Networking*, vol. 14, no. 3, June 2006.

[8] P. Indyk and J. Matousek, "Low-distortion embeddings of finite metric spaces," in *CRC Handbook of Discrete and Computational Geometry*, 2004, chapter 8.

[9] D. K. Agrafiotis and H. Xu, "A geodesic framework for analyzing molecular similarities," *J. Chem. Info. Comput. Sci.*, vol. 43, pp. 475–484, 2003.

[10] L. Tang and M. Crovella, "Virtual landmarks for the internet," in *Proc. ACM Sigcomm*, 2003, pp. 143–152.

[11] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," in *Computer Communication Review*, vol. 34, 2004, pp. 15–26.

[12] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic routing without location information," in *Proc. ACM Mobicom*, 2003, pp. 96–108.

[13] R. Fonseca, S. Ratnasamy, J. Zhao, C. T. Ee, D. Culler, S. Shenker, and I. a. Stoica, "Beacon-vector routing: Scalable point-to-point routing in wireless sensor networks," in *NSDI*, 2005.

[14] J. Newsome and D. Song, "Gem: graph embedding for routing and data-centric storage in sensor networks without geographic information," in *Proc. ACM SenSys'03*, Los Angeles, USA, 2003.

[15] J. Bruck, J. Gao, and A. Jiang, "MAP: Medial axis based geometric routing in sensor networks," in *Proc. ACM Mobicom*, Koeln, Germany, 2005, pp. 88 – 102.

[16] Q. Fang, J. Gao, L. J. Guibas, V. de Silva, and L. Zhang, "Glider: gradient landmark-based distributed routing for sensor networks," in *IEEE Infocom*, Miami, FL, Mar. 2005.

[17] Y. Shang, W. Ruml, Y. Zhang, and M. Fromherz, "Localization from connectivity in sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 11, p. 961, 2004.

[18] J. Kleinberg, A. Slivkins, and T. Wexler, "Triangulation and embedding using small sets of beacons," in *FOCS*, 2004, pp. 444–453.

[19] I. Abraham, Y. Bartal, T.-H. Chan, K. Dhamdhere, A. Gupta, J. Kleinberg, O. Neiman, and A. Slivkins, "Metric embeddings with relaxed guarantees," in *46th IEEE Symposium on Foundations of Computer Science*, 2005.

[20] T. Moscibroda, R. O'Dell, M. Wattenhofer, and R. Wattenhofer, "Virtual coordinates for ad hoc and sensor networks," in *Proc. DIALM-POMC*, Philadelphia, PA, USA, Oct. 2004.

[21] D. Tschopp, S. Diggavi, M. Grossglauser, and J. Widmer, "Robust Routing for Dynamic Wireless Networks Based on Stable Embeddings," in *Proc. Information Theory and Applications workshop (ITA)*, San Diego, CA, January 2007.

[22] K. A. Dimitris, "Stochastic proximity embedding," *Journal of Computational Chemistry*, vol. 24, no. 10, pp. 1215–1221, 2003, 10.1002/jcc.10234.