Topological Hole Detection in Wireless Sensor Networks and its Applications

Stefan Funke *
Computer Science Department
Gates Bldg. 375
Stanford University, CA 94305, U.S.A.

ABSTRACT

The identification of holes in a wireless sensor network is of primary interest since the breakdown of sensor nodes in a larger area often indicates one of the special events to be monitored by the network in the first place (e.g. outbreak of a fire, destruction by an earthquakes etc.). This task of identifying holes is especially challenging since typical wireless sensor networks consist of lightweight, low-capability nodes that are unaware of their geographic location.

But there is also a secondary interest in detecting holes in a network: recently routing schemes have been proposed that do not assume knowledge of the geographic location of the network nodes but rather perform routing decisions based on the *topology* of the communication graph. Holes are salient features of the topology of a communication graph.

In the first part of this paper we propose a simple distributed procedure to identify nodes near the boundary of the sensor field as well as near hole boundaries. Our *hole detection* algorithm is based purely on the topology of the communication graph, i.e. the only information available is which nodes can communicate with each other. In the second part of this paper we illustrate the secondary interest of our hole detection procedure using several examples.

Categories and Subject Descriptors: C.2.2 [Computer-Communication Networks]: Network Protocols

General Terms: Algorithms, Design

Keywords: Routing, Graph theory, Embedding, Virtual Coordinates, Topology

1. INTRODUCTION

Picture the following scenario: during a long summer drought, wild fires have started in a large region of a remote nature preserve that is hardly accessible by ground transportation. To be able to continuously assess the situation and plan appropriate countermeasures, airplanes are sent out to deploy thousands of wireless sensor nodes. Due to cost restrictions and to achieve the maximum life-time by energy savings, these sensor nodes are rather low-capability devices equipped just with temperature and humidity sensors, a simple processing unit, and a small radio device that allows for communication between nearby sensor nodes. One of the first goals is now to have this network organize itself such that messages are routed within the network, regions of interest (e.g. the current firefront) can be identified, and gathered data can be efficiently queried.

Achieving this goal becomes quite challenging since the only information a node has about the global network topology are its immediate neighbors with whom it can communicate. Lacking an energy-hungry GPS unit and being deployed from an airplane in a rather uncontrolled fashion, none of the sensor nodes is aware of its geographic location.

Before we describe our contribution in more detail, let us make the scenario more precise. Assume the area of interest is some region $\mathcal{R}$. The airplanes have deployed sufficiently many sensors such that – if all of the sensors were in operation after reaching the ground – the area of interest is completely monitored by the sensors. Formally we have that for every point $p \in \mathcal{R}$ there would be at least one sensor s within distance $d(p, s) \leq r_{\text{sense}}$. Where r_{sense} is the *sensing radius* of the wireless nodes, i.e. the radius within which they can monitor or estimate temperature or humidity. Unfortunately, not all sensors will be operational upon reaching the ground. Some of them might fall right into the flames and be destroyed, others might plunge into a lake or pond and be unable to perform their monitoring task. Paradoxically we are particularly interested in those areas where there's an ongoing fire (and maybe also where there is a lake or pond), but sensor nodes that fell into these areas are unable to report this fact to us.

Our approach of detecting the (boundaries of) such holes in the monitored space created by fire or other phenomena will based on the examination of the *communication graph* of the wireless nodes. The *communication graph* of a wireless network has a node for each wireless station and an edge between two nodes if the respective stations can communicate with each other. For simplicity let us assume that two nodes can communicate with each other if they are within distance of at most r_{comm} (communication radius). So the communication graph is essentially a *unit disk graph* (UDG). Typically, the communication radius r_{comm} is considerably larger than the sensing radius r_{sense}, let $\kappa = r_{\text{comm}}/r_{\text{sense}}$

*supported by the Max Planck Center for Visual Computing and Communication (MPC-VCC) funded by the German Federal Ministry of Education and Research (FKZ 01IMC01).

DIALM-POMC'05, September 2, 2005, Cologne, Germany.

be the ratio between these two quantities. Clearly, the larger the value κ becomes, the denser the communication graph gets. For example, a simple calculation shows that for $\kappa = 5$, i.e. the communication radius is five times the sensing radius – which is a quite realistic assumption for temperature, humidity, or acoustic sensors –, the degree of a node not close to a (hole or outer) boundary is at least 24.

In essence the primary problem that we consider in this paper is that of identifying holes just by examining a rather dense communication graph. If in a sufficiently large region sensors break down, this hole will also manifest itself in the communication graph, hence holes identified using the communication graph are indicative of some large-scale special event in the region to be monitored.

The secondary benefit of our hole finding routine arises more generally in wireless networks. Wired networks usually have a rather static behavior where nodes or links change seldom, and hence efficient routing can be achieved by aggregating routing information in IP addresses and by having powerful routers maintain these tables to distribute message within the network. Wireless networks on the other hand tend to be much less structured due to the volatility of node links. A lot of research effort has been spent in designing good protocols for this setting, see e.g. [14] for an overview.

One powerful approach to routing within wireless networks is the idea of *geographic routing* pioneered in GPSR / GFG [2, 7] and improved later e.g. in [8]. This approach is based on attaching geographic location information to each network node and then apply a greedy-routing mechanism where a packet is always sent to a neighbor which is closer to the final destination. The main disadvantage of these protocols is the need for location information for each node of the network. This can be, of course, provided by a GPS system on each network node, but given that in many scenarios especially in wireless sensor networks, the network nodes are very light-weight and low-capability devices, this is a rather prohibitive assumption. Alternatively, few dedicated nodes could be equipped with a GPS system and protocols for localization could be used to construct location information for the other nodes (see e.g. [4]).

More recently, several papers have proposed routing schemata which do not rely on geographic location information at all. They proceed either by constructing *virtual coordinates* as in [11, 13, 15] which are derived from the network topology but do not necessarily reflect the true geographic locations, or by routing via landmarks as in [12]. In spite of or maybe because they intentionally ignore any geographic information, these algorithms rely even more on other means of capturing the topology of the underlying network. The secondary benefit of the hole finding approach presented in this paper is to provide a powerful tool that exhibits more of this topology information to location-free routing protocols. In general, global topological features of the network like large holes tend to be stable under changes of few network links, so schemes based on these global topological properties, are likely to enjoy a natural robustness against network volatility, as it was for example shown in the GLIDER approach (see [12]).

Related to our work are geometry-based hole detection algorithms like [5] as well the statistics-based approach presented in [6]. They are not direct competitors to our approach, though, since the former inherently relies on the availability of geographic location information at the nodes, and the latter seems feasible only for uniform and very high density node distributions. We are not aware of other work addressing the same problem as our algorithm.

1.1 Our Contribution

In this paper we present a simple procedure to identify nodes on the outer and hole boundaries in a field of wireless network nodes. Our algorithm relies purely on the connectivity information of the underlying communication graph and does not make use of any location information of the nodes. While our work does not propose new routing protocols, we believe our procedure is of great interest to location-free algorithms that have recently become the focus of active research. We give evidence for the usefulness of our approach by providing two example applications where topological hole detection allows for improved performance of the existing schemata. In the first example we investigate a recent protocol called GLIDER [12], which proposes load-balanced routing in the absence of geometric location information by using so-called *landmarks* to give structure to the underlying network. A key issue of this approach – which was not addressed in the original paper – is the appropriate selection of landmarks as it immediately affects the performance of the network. Using our hole detection algorithm we provide a sound landmark selection strategy which guarantees certain desirable properties. Furthermore we investigate the use of our topological hole detection routine in the context of virtual coordinate computations as proposed for example in [13]. We show that using the additional information about hole boundaries, it is possible to actually approximate the original geometry information rather well from just the connectivity of the communication graph – even in the presence of holes. The distributed complexity of our algorithm is essentially the same as that of flooding the network a constant number of times (where the constant is really small!), and since we expect this type of topology-exploration to happen rather infrequently during the lifetime of the network, we believe that our approach can be very useful in practice.

2. TOPOLOGICAL HOLE FINDING

This section introduces a novel way to detect holes in networks consisting of a set of wireless nodes. While the intuition behind our approach stems from geometric observations, the resulting algorithm is purely based upon the topology of the connectivity graph and does not assume any geographical location information.

2.1 The continuous case

Picture the following continuous variant of our problem: Given a (possibly non-simply) connected region $R \subset \mathbf{R}^2$ and some point $p \in R$ – we call that point *beacon* –, we consider the isolevels of the *geodesic distance* function d_p from p in the domain R as in Figure 2.1. That is, for any point $x \in R$ $d_p(x)$ denotes the minimum Euclidean length of an open curve $\Gamma \subset R$ with one endpoint being p, the other x. Or in other words, $d_p(x)$ is the length of the shortest path from p to x which stays within R and avoids all holes.

The *isolevel, isoline*, or *contour* of level k of the distance function d_p is the set of points $I(k) = \{x \in R | d_p(x) = k\}$. In Figure 2.1 we have depicted the contours of level $10, 30, 50, \ldots$. If the region R is free of holes and all points on the boundary of R can be seen from p (without obstruction by a hole), the contour of level k is a subset of the

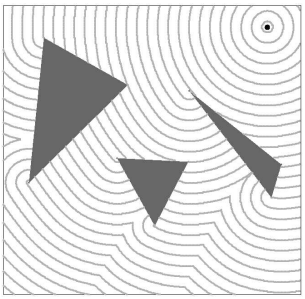


Figure 1: The continuous case: One beacon, 3 triangular holes, induced contours/isolines.

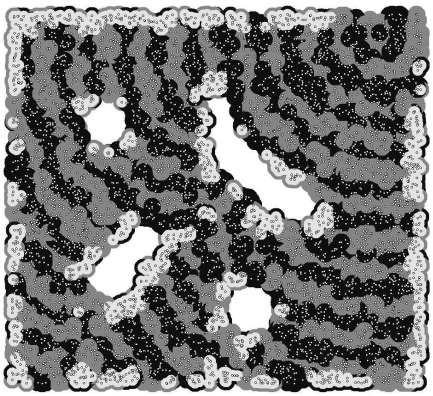


Figure 2: The discrete case: One beacon, its induced isolevels, and their marked boundaries

circle centered at p with radius k. In the more general case with polygonal obstacles, though, the contour of level k is a collection of (possibly disconnected) circular arcs.

What is interesting for our purposes is the observation that a contour is always 'broken' either at the outer boundary or the boundary of a hole. In fact the reverse, i.e. every point x of a hole/outer boundary is the endpoint of a contour component, is 'almost' true. The only way that x with $d_p(x) = k$ might not be the endpoint of a component of the contour of level k is in case that the tangents of the contour of level k and of the boundary agree at x. In Figure 2.1, this is the case for example in the right lower and left upper corner of the region R, where the 'wavefronts' hit tangentially the outer boundary. The same happens at the upper left tip of the rightmost triangular hole in the picture.

The key idea of our hole detection routine is to make use of these 'broken isolines' to determine nodes that are close to the outer boundary or to a boundary of some hole.

2.2 The discrete case

Of course, in the context of wireless networks, things are quite a bit different from the continuous setting. First of all, distances are not defined between any two points of the region R, but only between sensor nodes – hence the notion of contour curves and their breakpoints is completely ill-defined. Secondly, these distances are measured in terms of the *network distance*, i.e. the number of hops in the communication graph. In our scenario – where the network consists of a number of *light-weight* wireless nodes – there is no knowledge of geographic position, so these network distances are the only description of the relationship of a node to the rest of the network.

2.2.1 The Boundary-Detection Algorithm

Things are not hopeless, though. Given some beacon node of the network, what we can do, is to compute shortest distances in terms of hops in the communication graph of the network. And while there is no continuous notion of contour, we can define a similar structure via connected components of nodes with the same distance label. Assuming that within the 'non-hole' parts of R, the node distribution is reasonably high, the hop-distance between two nodes approximates geodesic distances reasonably well, and hence we can hope to mimic the continuous setting. We fill in the details in the following.

Before we give a precise description of our algorithm let us start with some definitions. As in the continuous setting, we have one dedicated network node p which is called the *beacon*. For every node v of the network we define as $d_p(v)$ the shortest (in terms of number of edges) path in UDG from p to v. For fixed integer k, we call $\mathcal{I}(k) = \{v : d_p(v) = k\}$ the isoset of level k. The subgraph of UDG induced by $\mathcal{I}(k)$ might be disconnected, so we call each of the resulting components of $\mathcal{I}(k)$ as $\mathcal{C}_1(k), \mathcal{C}_2(k), \ldots$. As in the continuous case we are interested in finding nodes near the points where the 'contours' are broken. It's just that in the discrete setting there are no contours but only connected components. Still, we can assume that when embedded into the plane at their true geometric position, a connected component of an isoset has a skinny, longish shape. So, if we pick some random node r (a *local beacon*) in $\mathcal{C}_i(k)$ and compute shortest-hop distances restricted to $\mathcal{C}_i(k)$, we expect the highest distance values at both 'ends' (this refers to their location when embedded in the plane). We mark these nodes with highest distance values as being on the boundary.

See Figure 2 for the markers created by a run of this procedure from a beacon in the top right corner. We have illustrated isolevels by black and dark grey colors, the marked nodes are distinguished by light grey disks around them. Observe that as predicted for the continuous case, boundaries hit tangentially by an isolevel are marked rather unreliably (see in particular the parts of the holes closest to the beacon). But we can remedy this by just repeating the procedure with another beacon and taking the union of all marked nodes. In our experiments it turned out that repeating this procedure for 4 beacons yields very good results without increasing the running time too much. So the description of our algorithm looks as follows:

HOLE-FINDER()

1. pick 4 beacons p_1, p_2, p_3, p_4, amongst the network nodes
2. for each p_i do:
   (a) compute distances of all network nodes to p_i
   (b) for all $1 \leq k \leq \text{diam}(UDG)$:
   - determine components $\mathcal{C}_1(k), \mathcal{C}_2(k), \ldots$
   - for each $\mathcal{C}_j(k)$, determine boundary nodes by calling **CompBounds(**$\mathcal{C}_j(k)$

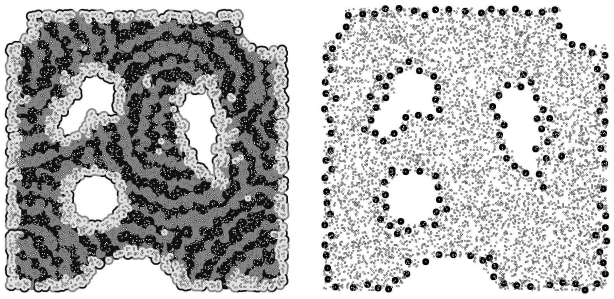


Figure 3: Output of algorithm after choosing 4 beacons and the resulting classification (left). Result after coarse boundary sampling (right).

The subroutine for exploring a connected component is implemented in a more conservative manner than described above. Instead of marking all nodes that do not have any 'higher' neighbor, we postulate that no 2-hop neighbor should be higher:

CompBounds ($\mathcal{C}$)

1. pick a local beacon q in $\mathcal{C}$
2. compute hop-distances $h'(v)$ to q in the subgraph induced by $\mathcal{C}$
3. mark all nodes $v \in \mathcal{C}$ which do not have a 2-hop neighbor $v' \in \mathcal{C}$ with $h(v') > h(v)$.

The only missing part is the beacon selection which we address in the following.

2.2.2 *Beacon selection*

How to pick the 4 beacons is of course quite important. If they are all close together, each of their induced distance fields looks about the same and no additional information/boundary markers are gained. A very simple strategy is just to pick 4 *random* nodes. Typically these are rather well-spread over the region and hence yield good results.

Another strategy is to *force* them to be wide-spread over the network. This can be done by having each node v of the network maintain a variable $CBD(v)$ (for Closest Beacon Distance) storing the (hop-)distance to the closest beacon processed so far. After processing beacons $p_1, \ldots p_{i-1}$, p_i is then picked amongst the nodes with maximal $CBD(.)$ value. This forces the beacons to be spread out over the network. In Figure 3 (left) we can see the result of our hole finding algorithm with 4 beacons. The first beacon was chosen randomly, the last 3 following the above strategy (they are located in top left, lower left, and lower right corner). Observe that both outer boundary as well as hole boundaries are rather well captured by a 'thickened' region of marked nodes around them. There are some spurious marked nodes in the middle of the network.

2.3 Coarse Boundary Sampling and Pruning

In particular for the applications discussed in the second part of this paper, it might be useful to get a more compact representation of the boundaries. Our hole finder algorithm essentially marks *all* nodes that are close to a boundary. The number of these nodes can be quite considerable, so a natural way to reduce this number is to compute a maximal independent set within all the marked nodes which can be computed using one of the standard distributed algorithms for the problem (see e.g. [10]). The result can be seen in Figure 3 (right). Furthermore, in this process we can also easily get rid of incorrectly marked nodes by only taking into account connected components of the marked nodes that have a certain minimum size (observe the missing spurious nodes in Figure 3 (right)).

2.4 Distributed Implementation and Efficiency

So far we have not mentioned anything about a distributed implementation of our algorithm. But as we expect that topology exploration has to be performed only rarely during the lifetime of the network (typically at the beginning, after deployment, and then only in rather large time intervals to adapt to topology changes), we can afford a straightforward, naive implementation. Essentially the cost of our algorithm is that of flooding the network with a message a constant number of times: computing the distance to a beacon can be implemented by flooding the network with a HELLO message originating from the beacon maintaining a distance counter in the message that is incremented at every hop. A node's distance to the beacon is then just the minimum counter value over all message received. Similarly the distance computations within each isolevel sum up to a flooding of the whole network. If beacons are not to be chosen randomly but using a deterministic method, this can also trivially be implemented using $O(n)$ messages and time. This, of course assumes handling of interference as well as knowledge of the one-hop neighborhood of each node. But this assumption on the existence of a low-level infrastructure is commonly accepted.

2.5 Summary

As we will see later in the experimental section, our hole finding approach turns out to work really well for scenarios where the network density is reasonably high. In contrast to other approaches like [6], we do not require an extraordinarily high node density – anything higher than average degree 16 for the communication graph seemed to work fine –, nor can our algorithm be fooled by changing node densities, as none of the algorithm's decisions are directly based upon the *number* of neighbors.

In Figure 4, we have generated points randomly in the $[0:1] \times [0:1]$ square and then replaced each x-coordinate by $x/2 + x^2/2$ and each y-coordinate by $y/2 + y^2/2$. The resulting set of points is still contained in the unit square, but somewhat biased towards the origin. Still, our algorithm faithfully determined the boundaries (in the Figure, the output after sampling and pruning is shown).

3. APPLICATION: TOPOLOGY-BASED LANDMARK-SELECTION

In [12] Fang et al. present a novel routing scheme named GLIDER which is purely based on the node-topology and does not require any geometric information. The key idea is to choose a set of distinguished nodes – so-called *landmarks* – and divide the network into *tiles*, that is connected subnetworks which have one particular landmark as closest in terms of the graph distance (they essentially mimic a geometric *Voronoi* diagram in the graph setting). The number

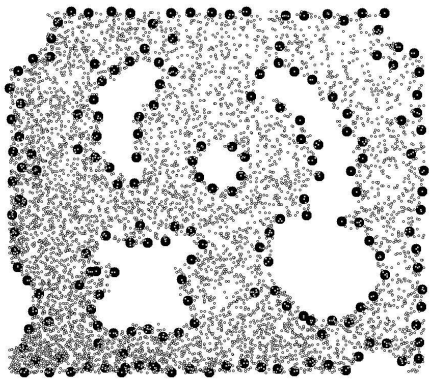


Figure 4: Complicated sensor field topology with non-uniform density

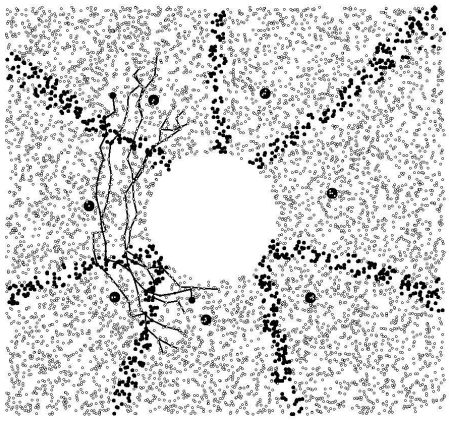


Figure 5: Natural load balancing in inter-tile routing

of landmarks is supposedly rather small compared to the network size, so *every* node in the network can afford to have knowledge about all landmarks, distance to the landmarks, and the adjacency information of landmark tiles (i.e. which tiles have common boundaries) – the authors call this the Landmark Voronoi Complex (LVC). Within each tile, a local coordinate system is constructed, and a network node is named by a two-tuple consisting of the tile where it resides and the local coordinates within that tile.

When a node p in tile t_p wants to send a message to a node q in tile t_q, using the connectivity information of the LVC stored at every node, p can determine a sequence of tiles $t_p = t_1 t_2 \dots t_q$ to get to tile t_q containing q. The routing strategy for the message on its way to tile t_q is as follows: when being in tile t_i the message is always sent to a neighbor that is closer to the landmark of tile t_{i+1}. When crossing the boundary to tile t_{i+1}, the 'target' is switched to t_{i+2} and so on. Note that this *inter-tile* routing scheme provides for a very natural and elegant way of load-balancing, see Figure 5. Even paths that share the same subsequence of tiles, are kept apart, as shown in the example where several paths were routed from random positions in the lower center tile to the upper left tile. We refer to [12] for a more detailed description of this scheme. Finally being in the tile t_q where q resides, a greedy procedure based upon the local coordinates is used to get to the destination q. If this fails due to a local minimum, the tile is flooded with the message.
We cite from the GLIDER paper [12]:

> While the definition and properties of LVC and CDT hold for any subset of landmarks, careful selection of landmarks is crucial for the effectiveness and efficiency of routing.

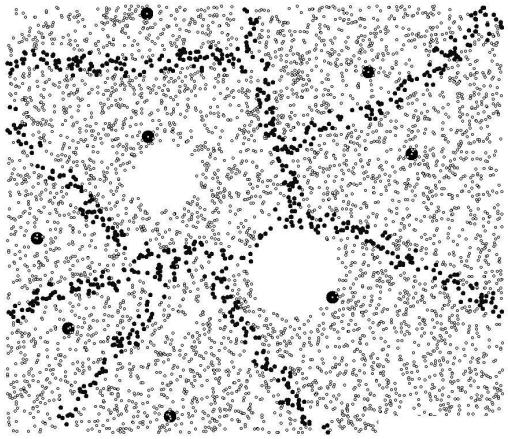
Fang et al. propose that the number of landmarks should be roughly proportional to the number of holes in the communication network. But even such information is typically not available a priori. We propose to use our hole finding algorithm to help with the landmark selection process. While just obtaining a rough count of the number of holes in the network obviously already helps improve the performance of GLIDER, we will also illustrate in the following how topology-unaware landmark selection can decrease network performance (even if a rough count of the number of holes is available). We then propose a landmark selection scheme based on the presence of holes. Again, we want to emphasize that without a hole detection algorithm not even a rough estimate of the number of holes in the network is available a priori.

3.1 Problems of unaware Landmark-Selection

In the following we will briefly illustrate the problems incurred by landmark selection that is unaware of the actual topology of the network. Figure 6 shows the LVC for two different landmark selections. On the left, a random set of 8 landmarks was chosen, on the right, the 8 landmarks were chosen taking into account the topology of the network (in fact based upon our hole finding algorithm). One can see that on the left, one of the main topological features of the network – one of the two holes – was essentially overlooked by the LVC, as it resides completely within a tile. This has implications for the efficiency of both inter-tile as well as intra-tile routing as we will see in the following. On the other hand, the topology-aware landmark selection ensured that all tiles are topologically simple. This is essential for the LVC routing strategy to perform as claimed.

Let us now turn to concrete examples where we can exhibit the implications of topology-unaware landmark selection. We first consider inter-tile routing. In Figure 5 we have seen that the LVC-based routing provides for a very natural load balancing mechanism, where packets – even though traversing the same sequence of tiles – are routed on different paths. Unfortunately this nice property does not hold anymore when holes are contained within a tile. Consider the situation in Figure 7. Here a hole contained within the center tile essentially forces all the routing paths to pass close to the hole boundaries. In the example we have generated random path queries from the lower center tile to the upper center tile; in absence of the hole, the load would have been rather nicely distributed.

Another problem arises in the intra-tile routing phase of the LVC protocol. As soon as a packet has arrived at the tile of the destination, a greedy gradient-based routing scheme is employed to find a path to the destination. In Figure 8, we have depicted the distance values to a destination (marked by a white cross slightly above the hole) within a tile according to the local coordinates and the distance measure as defined in [12]. Darker colors denote smaller distances.



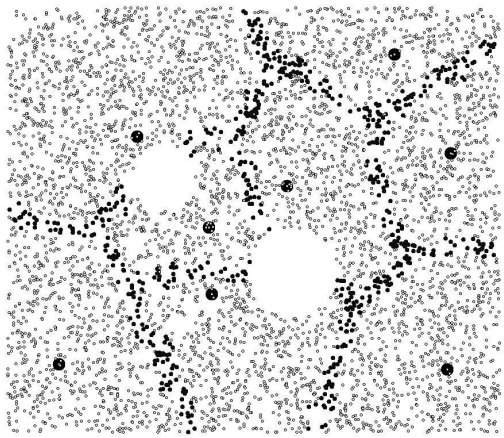


Figure 6: LVCs for different landmark selection: random vs. topology-aware

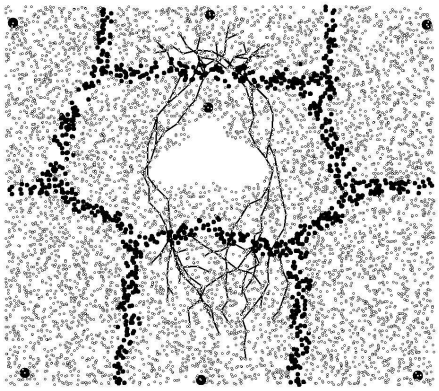


Figure 7: Uncaptured topology: hole contained within Voronoi tile and resulting load imbalance in inter-tile routing

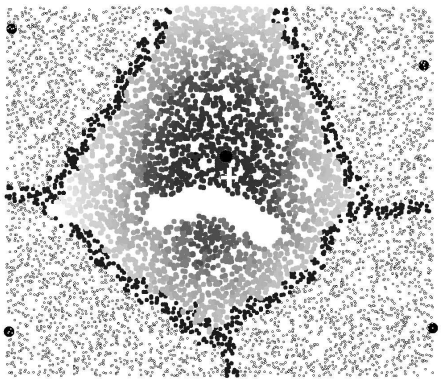


Figure 8: Uncaptured topology: hole contained within Voronoi tile and resulting local minima in intra-tile routing

As we can see, the presence of the hole creates a region with a local minimum (below the hole) where no descending path to the actual destination is possible. In this case, the LVC protocol resorts to flooding the whole tile with the message.

We summarize that it seems absolutely essential for the performance of the LVC routing protocol to ensure that the landmark selection takes into account the actual topology of the network, in particular holes should never be completely contained within a tile of the LVC.

In the following we will explain how to make use of our hole finding routine to obtain good landmark selections. As in case of the hole detection, the intuition is taken from the continuous case, but practically translates well to the discrete case.

3.2 First Attempt: Capturing Topology by Hole Sampling

The first attempt is motivated by the simple observation that if at least two landmarks are on the boundary of every hole, there cannot be a hole completely contained within a

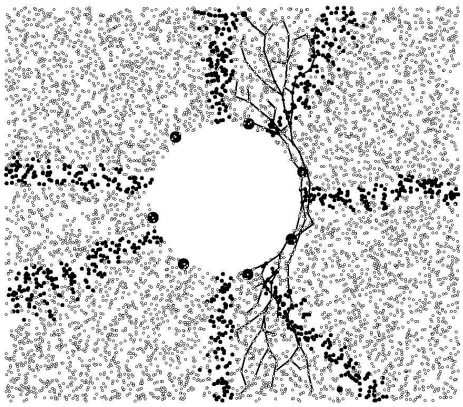


Figure 9: Load imbalance due to Landmarks being too close to boundaries

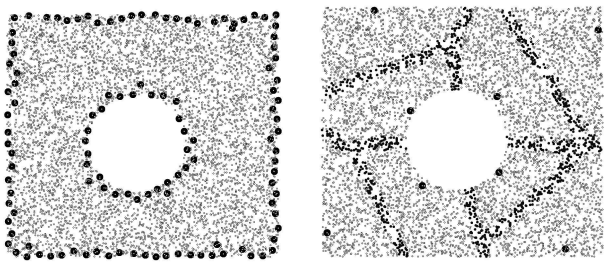


Figure 10: Coarse boundary (left) and result of feature-sensitive pruning (right, with resulting LVC)

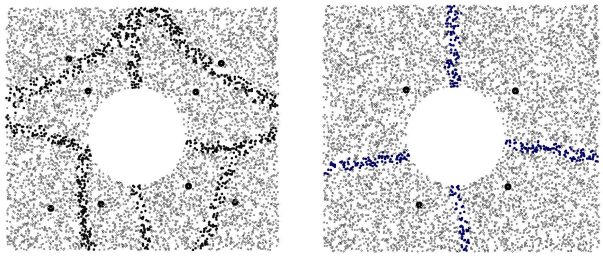


Figure 11: Landmark selection with hole repulsion (left) and pruning (right)

tile of the LVC. This is clearly true since hole containment implies that all nodes on the boundary of the hole have one single landmark as their closest landmark. We have modified our hole finding algorithm to decimate the sampled and pruned output (Figure 10, top) further by allowing only 4 marked nodes on the boundary of a large hole (we allow only 2 marked nodes on smaller holes). The respective landmarks and the resulting LVC can be seen in Figure 10, bottom. We have 4 landmarks on the boundary of the hole and 4 landmarks on the outer boundary.

What seems not so nice is the fact that the landmarks always lie quite off-center in their respective tile in the VC. This has some implications for the load balancing behavior of the LVC protocol, as can be seen in Figure 9. Here, due to the landmarks being too close to the boundary of the hole, all traffic from the lower center to the upper cell is attracted towards the hole boundary and hence load-balancing behavior gets quite bad. This is similar to the case depicted in Figure 7.

3.3 Second Attempt: Hole-Repulsion and Pruning

The idea for remedy is to have the holes 'repulse' the landmarks and by that allow for tiles with more centered landmarks. But we still want to ensure that no hole is completely contained within a tile of the LVC, which might happen if landmarks are moved too far away from the hole boundaries. Let us describe our strategy by focusing on one hole. To move landmarks 'away' from hole boundaries it is convenient to have each node v in the network store a value $\Delta(v)$, which denotes the hop-distance to the closest boundary node. This can easily be implemented by a one-time-flooding procedure where each sampled boundary node (before decimation to get the preliminary landmarks) sends a HELLO message with distance counter 0 which increases at every hop. The value $\Delta(v)$ is then the minimum counter value over all messages received.

For each landmark p chosen according to the previous strategy let $d_{\text{local}}(p)$ denote the minimum hop-distance to another landmark q of the same hole. We replace p by some node p' within p's $(d_{\text{local}}(p)/3)$-hop neighborhood which has maximal $\Delta(p')$ value (intuitively: which is furthest away from a boundary). Observe that moving all landmarks in this way, still ensures that no hole is completely contained within one tile. The reason for that being that p – the old landmark on the boundary of the hole – is still closer to p' than to any other landmark of the same hole (since they were at least $d_{\text{local}}(p)$ hops away and have moved by at most $d_{\text{local}}(p)/3$ towards p. So p will be in p''s tile whereas by the same argument node q of the same hole will belong to q''s tile. Hence this hole cannot be completely be contained in one tile. This procedure might move landmarks of different holes rather close together, creating unnecessary landmarks, see Figure 11, left. But there is a way to prune the landmarks even further – without violating the guarantee of hole-free tiles. Let us consider a pair of landmarks (p', q') (after the movement and not originating from the same hole) at hop-distance h. We claim that if $h < d_{\text{local}}(p)$, p' can be removed (if q' remains in the set of chosen landmarks). As before this can easily be verified by establishing that q' is closer to p than any other landmark originating from the same hole as p. The resulting set of landmarks can be seen in Figure 11, right. The landmark set of Figure 6, right, was also selected using this process of repulsion and pruning (without pruning, there used to more redundant landmarks in the center between the holes).

3.4 Distributed Implementation

As in case of the hole detection algorithm, we assume that the landmark selection process will be performed not very frequently. As already mentioned in [12], the communication cost of establishing the landmark Voronoi complex is essentially dominated by flooding the network k times, where k is the number of landmarks selected (according to our landmark selection scheme k will be in the order of the number of holes in the network). This also dominates the cost of computing hole boundaries and the topology-aware selection of landmarks.

4. FURTHER APPLICATIONS

In this section we will rather briefly sketch other applications where our hole finding routine might be of interest.

4.1 (Virtual) Coordinates in the absence of Location Information

In [13] Rao et al. have presented a scheme for computing virtual coordinates of network nodes based purely on the connectivity information of the communication graph. They first determine a set of nodes on the outer boundary of the sensor field Then using the distance information between all boundary nodes which are used as approximations of their Euclidean distances, they embed these nodes in the Euclidean plane by a simple triangulation scheme. (Virtual) coordinates of non-boundary nodes can then be obtained either by triangulation or by an iterated relaxation algorithm.

The problem with the straightforward extension of this

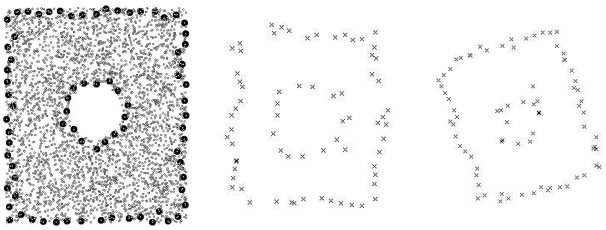


Figure 12: Node distribution in a sensor area with one hole (left), embedding of boundary nodes using standard triangulation (center), embedding only relying on 'truthful' distances (right)

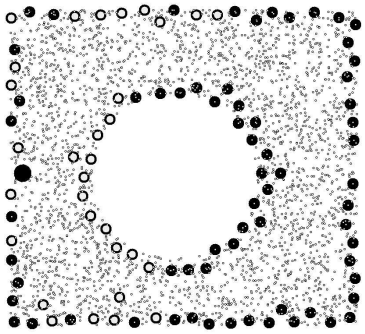


Figure 13: 'Truthful distances' from bold point

scheme is that in a region of non-simple topology, holes might 'obstruct' the shortest paths between nodes of the network and hence their lengths are not a good estimate of the true, geometric distance. As already observed in [13], this leads to an exaggerated blow-up of the size of contained holes, see Figure 12, center.

4.1.1 Embedding via Triangulation on 'truthful' Distances

One way to solve this problem is to 'classify' distances based on the fact whether in their measurement via hop-distances in the communication graph, the shortest path got diverted due to holes in the network. We do this as follows: let P be the set of boundary nodes that are to be embedded. We say the distance measured between a pair $(p, q) \in P \times P$ is *truthful*, if the respective shortest path in the communication graph from p to q providing this estimate does not contain any $r \in P$ as intermediate node. The intuition behind that is, that non-obstructed nodes certainly do not have another boundary node on their shortest connecting path, whereas in the presence of holes, chances are quite good that one of the marked boundary nodes of the obstructing hole are part of the shortest path. See Figure 13 for the nodes which have truthful distances (the hollow nodes) to a fixed node (bold).

In our embedding algorithm we then start with the distance matrix and another matrix with boolean entries specifying whether a distance between two nodes is truthful or not. We start by picking a set of 3 well-spread, but mutually truthful set of nodes (preferably three nodes that form a non-skinny triangle) and embed them in the plane (the first point at the origin $(0, 0)$, the second at $(0, d_{12})$, and the third at either intersection of the circle centered at $(0, d_{12})$ of radius d_{23} and the circle centered at $(0, 0)$ of radius d_{13}. From now on we pick the additional nodes one by one and determine their location in the embedding by a simple triangulation scheme. A new node p with truthful distances to already embedded nodes q_1, q_2 has two potential locations (again the two intersection points of the respective circles centered at the locations of q_1 and q_2). To choose the correct one of these two, we select another already embedded node q_3 which is close to either of the intersection points. We iterate this procedure until all nodes have been embedded. Of course, quite important for the quality of the outcome is a 'good choice' of the order of the nodes to be embedded and which already embedded nodes to base the new location upon. Again as for the first three points, we prefer nicely-shaped, 'fat' triangles, and direct distances between the nodes involved. The result of this procedure can be seen in Figure 12, right, where – like the direct approach – the global topology is roughly captured, but the extensions of the hole are far more accurate due to the more conservative use of distances only if they are 'truthful'.

4.2 Medial-Axis-Based Routing

In a very recent work ([3]), a routing scheme based on the *medial axis* of the underlying domain has been developed. The *medial axis* M of a set of points $P \subseteq \mathbf{R}^2$ is defined as the locus of all points which have more than two closest neighbors in P. For a region R bounded by a closed curve potentially with holes described by other closed curves, we are interested in the part of the medial axis of these curves, that lie within the interior of R. The approach described in [3] derives for each node of the network canonical coordinates which are then used for all routing decisions. While the medial axis at first sight is a purely geometric structure (in some sense the continuous version of the Voronoi diagram), our hole finding algorithm can be used to simulate it even in the absence of any geometric information at the nodes. The idea is first to determine boundary nodes and then compute for every node of the network its distance to all hole boundaries (again this can be done by sending HELLO messages from the hole boundaries maintaining a distance counter). In case of convex holes, a node is then part of the medial axis / the Voronoi diagram if its two 'closest holes' are at about the same distance. For non-convex holes things get slightly more difficult, see [3].

In Figure 14 we have determined for each node in the network the distance to the closest boundary node (constructed by our hole finding algorithm) – light color denotes large, dark color denotes small distance values. We see that the resulting picture resembles the (geometric) Voronoi diagram and hence the medial axis without actually using any geographic location information.

5. EXPERIMENTAL EVALUATION

Our algorithm as well as the applications mentioned in the second part of the paper have been implemented and evaluated on various problem instances. While our implementation does not mimic network attributes like packet loss or delay, we are confident that the results give a good indication about the usefulness of our approach in practical scenarios. Due to space restrictions we focus in this section on the performance of the hole finding algorithm itself and leave the evaluation of the applications where the algorithm

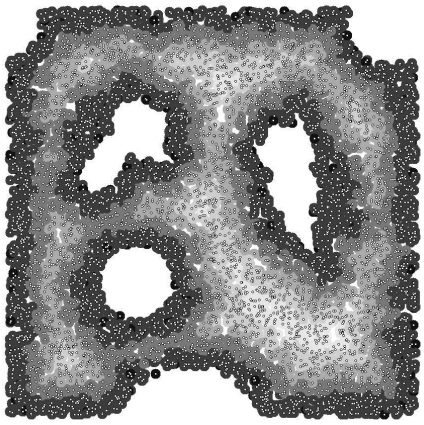


Figure 14: Approximate offsets from hole-detection for medial-axis approximation

is employed as a subroutine for the long version of the paper. The benchmark communication graphs we use for our experiments contain several thousand wireless nodes, which might seem overly large. But in recent research projects like ExScal ([1]) people have actually started operating wireless sensor networks whose sizes are in the thousands.

5.1 Unit-disk graphs

We examine the performance of our algorithm on classical unit-disk graphs, i.e. we assume that two nodes can communicate with each other if they are closer than the global communication radius.

5.1.1 Random Uniform Distributions

In the first experiment we placed 4900 nodes in a 800×800 square region uniformly at random and then removed a disk-like center portion of the nodes. By adjusting the communication ranges between 15 and 40 we obtained communication graphs of different densities. The pictures in Figure 15 show the results after running our hole detection algorithm (including coarsening and pruning). In parentheses we state the average degree of the respective communication graph. It turns out that only for graph densities with average degree around 18 or more our hole finding algorithm produces reasonable results. For lower densities, the output is of no use. We note, though that due to well-known transition-phase results, the graph with communication radius 15 and respective avg. node degree of 5 is hardly connected, i.e. even the largest connected component is very small. For avg. node degree of 10 the UDG is still not connected, though there is a rather large connected component.

5.1.2 Randomly perturbed Grid

Several routing papers evaluate their routing protocols on networks generated by a randomly skewed grid. This is a reasonable model in case the wireless nodes were e.g. deployed by an aircraft in a roughly regular pattern, but due to imprecision in the deployment process, the wireless nodes do not end up exactly at desired location. We placed 4900 nodes on a 70×70 grid with grid width of about 11. We then perturbed each grid point in x- and y-direction by a random amount between 0 and 11. Visually the node distribution looks similar to the random distribution, but the resulting communication graph is much better connected. So – like many other routing schemes – our algorithm performs extremely well, faithfully capturing the hole even in networks of average degree 10. See Figure 15 for the output.

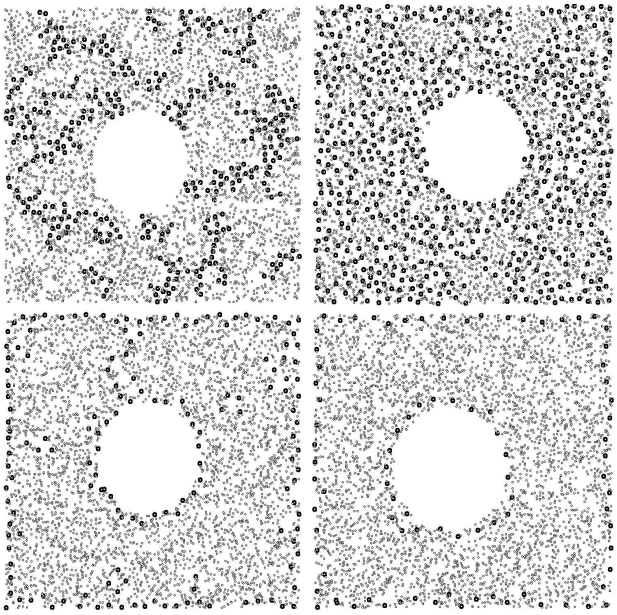


Figure 15: Algorithm output for UDGs of *randomly distributed points* and communication ranges 15 (avg. degree 5), 20 (10), 27 (18), 40(39)

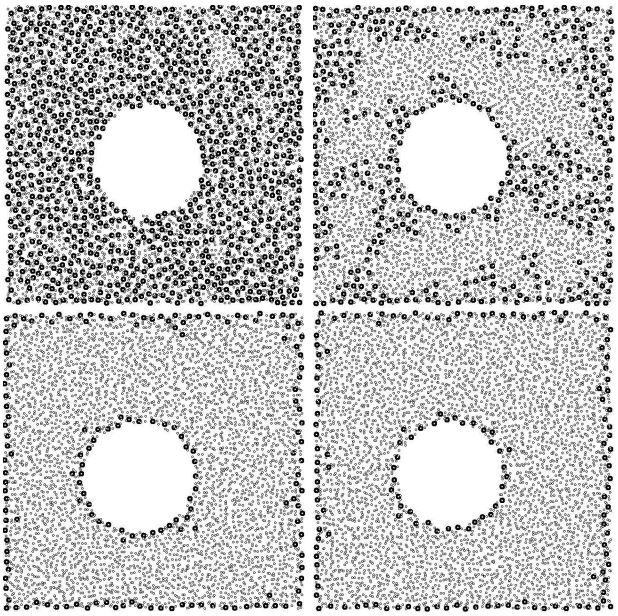


Figure 16: Algorithm output for UDGs of *skewed grid points* and communication ranges 15 (avg. degree 5), 17 (7), 19 (10), 20(11)

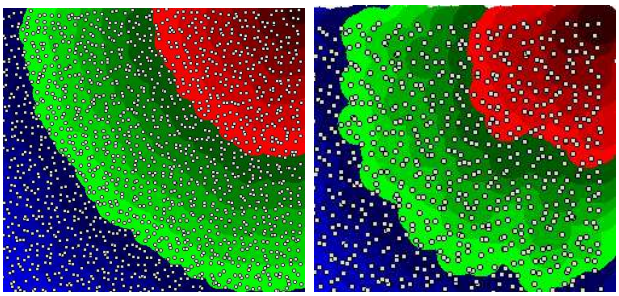


Figure 17: Wavefronts for a unit-disk graph (left) and a non-unit-disk graph (right).

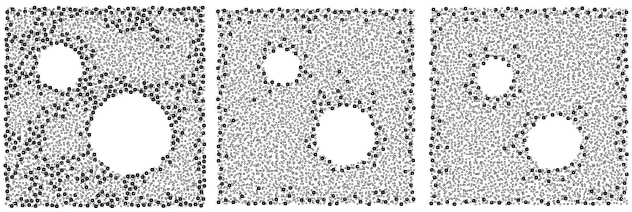


Figure 18: Algorithm output for non-UDGs of average degrees 8,16, and 20.

5.2 Non-UDGs

Our algorithm does not crucially rely on the graph being a unit-disk graph, but also tolerates deviations e.g. inspired by the so-called quasi-unit-disk-graph [9]. In our model we have a lower bound r_{low} as well as an upper bound r_{up} for the communication radius. Two nodes at distance at most r_{low} can *always* communicate with each other whereas for nodes at distance between r_{low} and r_{up}, they can only communicate with a certain probability p. In Figure 17 we have depicted the distance fields induced by a 'regular' unit-disk graph (left) as well as a non-unit-disk graph according to our model with a probability $p = 1/20$ of edges of length between r_{low} and $r_{\text{up}} = 2 \cdot r_{\text{low}}$ being present in the communication graph. While the general appearance of a wavefront remains the same, this non-unit-disk graph model leads to less 'smooth' distance fields. We note that increasing the probability p actually smoothes out the wavefronts.

The success rate of the hole detection decreases with the volatility of the communication radius. See Figure 18 for detection results in our non-UDG graph for varying average degrees 8, 16, and 20. For low average degrees in the non-UDG, the performance is comparable to the UDG case, though increasing the average degree does not lead to the very 'clean' results as in case of the UDG. The sensor nodes were placed on a perturbed grid as in Figure 16. What is important to note here, though is the fact that the hole detection algorithm still does reasonably well as it does not rely on the hop-distances in the graph resembling closely the geometric distance between the respective nodes; crucial is rather the fact that nodes with the same hop-distance label form a connected wavefront.

6. CONCLUSIONS

In this paper we have presented a rather simple and straightforward algorithm for detecting holes in a wireless communication network, that is purely based on the connectivity of the communication graph. This has immediate applications in the interpretation and evaluation of data acquired via a wireless sensor network, since large holes in the communication graph are typically caused by events that the sensor network was installed to monitor in the first place. We have also sketched further applications of our hole finding routine, where the knowledge about holes in the network provides for better performance of existing topology-based, location-free protocols. Currently, a lot of research is conducted in the area of location-free routing and organization of networks. The reason for that is the intuition that salient, global features like large holes in the communication graph are rather stable under local changes in the network topology. Hence, algorithms or protocols relying on this global topology are likely to enjoy a certain natural robustness against local network changes. We believe that hole detection will play a crucial role in this context.

While our experiments suggest that the presented hole detection approach is viable only in the setting of large, rather dense communication graphs, recent deployments of wireless sensor networks have reached such densities and sizes of several thousand nodes, see for example the ExScal project ([1]. We expect a rapid increase in the deployment of large-scale sensor networks in the near future and a growing interest in topology-based routing and organization schemes, since equipment of such a large number of nodes with location devices like GPS is prohibitive due to its cost.

7. REFERENCES

[1] A. Arora, R. Ramnath, P. Sinha, and et al. Project exscal. In *Proc. 1st Int. IEEE Conf. on Distributed Computing in Sensor Systems (DCOSS)*, 2005.

[2] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7(6):609–616, 2001.

[3] J. Bruck, J. Gao, and A. Jiang. Map: Medial-axis-based geometric routing in sensor networks. Proc. *MobiCom*, 2005.

[4] D. De Couto and R. Morris. Location proxies and intermediate node forwarding for practical geographic forwarding, *MIT-LCS-TR824*, 2001.

[5] Q. Fang, J. Gao, and L. Guibas. Locating and bypassing routing holes in sensor networks. In *23rd Conf. of the IEEE Communications Society (INFOCOM)*, 2004.

[6] S. P. Fekete, A. Kröller, D. Pfisterer, S. Fischer, and C. Buschmann. Neighborhood-based topology recognition in sensor networks. In *ALGOSENSORS*, 2004.

[7] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Mobile Computing and Networking*, pages 243–254, 2000.

[8] F. Kuhn, R. Wattenhofer, and A. Zollinger. Asymptotically optimal geometric mobile ad-hoc routing. In *Proc. DIAL-M*, 2002.

[9] Fabian Kuhn and Aaron Zollinger. Ad-hoc networks beyond unit disk graphs. In *Proc. DIALM-POMC*, pages 69–78, 2003.

[10] T. Moscibroda and R. Wattenhofer. Maximal independent sets in radio networks. In *24th ACM Symp. on the Principles of Distributed Computing (PODC)*, 2005.

[11] Radhika Nagpal, Howard E. Shrobe, and Jonathan Bachrach. Organizing a global coordinate system from local information on an ad hoc sensor network. In *Proc. IPSN*, 2003.

[12] L. Guibas Q. Fang, J. Gao, V. de Silva, and L. Zhang. Glider: Gradient landmark-based distributed routing for sensor networks. In *24rd Conf. of the IEEE Communications Society (INFOCOM)*, 2005.

[13] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica. Geographic routing without location information. In *Proc. MobiCom*, 2003.

[14] E. Royer and C. Toh. A review of current routing protocols for ad-hoc mobile wireless networks. In *IEEE Personal Communications*, 1999.

[15] Y. Shang, W. Ruml, Y. Zhang, and M. Fromherz. Localization from mere connectivity. In *MobiHoc'03*, 2003.