

Chapter 5

Geometric Routing in Wireless Sensor Networks

Jie Gao

Abstract This chapter surveys routing algorithms for wireless sensor networks that use geometric ideas and abstractions. Wireless sensor networks have a unique geometric character as the sensor nodes are embedded in, and designed to monitor, the physical space. Thus the geometric embedding of the network can be exploited for scalable and efficient routing algorithm design. This chapter starts with geographical routing that use nodes' geographical locations to guide the choice of the next hop node on the routing path. The scalability of geographical routing motivates more work on the design of virtual coordinates with which greedy routing algorithms are developed and applied to route messages in the network. The last section is concerned about data-centric routing, in which a query is routed to reach the sensor node holding data of interest. Thus the challenge is to discover the “source node” that possess the data as well as route the message there.

5.1 Introduction

The eternal goal and the most fundamental problem for any types of networks are to enable efficient information delivery between the peers. A routing protocol, which establishes routes between a pair of nodes efficiently and correctly so that messages can be delivered in a timely manner, is an important component of the network architecture.

Traditional Internet routing achieves scalability by address aggregation, in which the routes to multiple destinations are summarized by a single routing table entry. In multi-hop sensor networks, this is not applicable as the node IDs of nearby sensor nodes are by no means close to each other. However, wireless sensor networks have a unique geometric character as sensor nodes are embedded in, and designed to

J. Gao

Department of Computer Science, Stony Brook University, Stony Brook, NY 11794, USA
e-mail: jgao@cs.sunysb.edu

monitor, the physical environment. Thus the physical locations of sensor nodes have provided a lot of opportunities to be exploited for efficient and scalable routing mechanisms in sensor networks.

This chapter surveys two topics in which geometric ideas are used extensively for routing in wireless sensor networks: *point-to-point routing* and *data-centric routing*.

Point-to-point routing is to find routes between source and destination efficiently. Because of the limited resources available at sensor nodes (communication, power, and memory constraints), it is important to have a light-weight routing protocol with low maintenance effort and small state information, in order to be scalable. The set of routing protocols covered in this survey achieves scalability by using the geographical locations of the sensor nodes, or virtual coordinates (aka. names) assigned to the nodes, with which routing is often conducted in a local and greedy manner.

The second category of geometric routing focuses on *data-centric routing*, i.e., routing to find the data of interest. This is motivated by the application-specific characteristics of sensor networks and that the node itself is not as interesting as the data it senses and delivers. In data-centric routing, a query carries a description of the data types it is looking for and the routing protocol routes the query to meet the data.

At last we will cover *location service and hierarchical routing* in sensor networks. Location service is an important supporting infrastructure component for point-to-point routing with geographical coordinates or virtual coordinates. Each sensor node often comes with an ID, assigned during manufacture, which uniquely identifies the node but is typically not used to aid routing. A point-to-point routing scheme often assigns a *name* or *routing address* to a node to which a routing protocol is applied. In geographical routing, for example, the geographical location of a node is its routing name. A source routes to a destination with the name of itself and the name of the destination. Location service is to maintain the mapping of IDs and routing names and support queries for node names. Location service can be considered as a special case of data-centric routing, in which the “data of interest” is the name of the destination given its ID.

This chapter focuses on the design principle and the main idea in each algorithm. Please refer to the original papers for implementation details and experimental evaluations. The typical setting is a large homogeneous sensor network with nearby nodes being able to communicate directly and far away nodes using multi-hop routing to communicate with each other. Since many of the routing protocols use the sensor locations, we first survey background knowledge on localization protocols to find the locations of the sensor nodes.

5.2 Background

Sensor location information is indispensable for both sensor data integrity and network organization. Traditional approaches to obtain location information include global positioning systems (GPS) [49]. But GPS is not appropriate for large-scale

sensor network localization because of its high cost, large form factor, and outdoor constraints. There has been a lot of study on localization algorithms that derive the locations of sensor nodes from local measurements including distance and angle estimations between neighboring nodes [7, 23, 44, 71, 75–77, 86, 87, 89, 91]. The distance between two communicating nodes can be estimated by received signal strength indicator (RSSI) or time of arrival (ToA) techniques. Angles between adjacent edges can be measured by multiple ultrasound receivers [80], or directional antennas and laser transmitters/receivers.

Generally speaking, localization algorithms can be classified as anchor-based and anchor-free methods. Anchor-based methods assume that a (sometimes large) number of anchor nodes know their positions already [23, 75–77, 86, 87, 89]. The rest of the sensor nodes derive their positions by using distance or angle measurements to anchor nodes. In anchor-free methods, no node knows their absolute coordinates. The output is the relative positioning of the sensors, subject to a global translation and rotation.

A number of anchor-based localization algorithms use iterative trilateration or its variants. Anchor nodes obtain their locations by GPS or as prespecified. Then the other nodes find their locations by trilateration progressively [75, 86, 87]. In the basic trilateration scheme, a node p can determine its location with distance measurements to three anchor nodes that are not colinear. Then node p becomes a new anchor node. See Fig 5.1. Repeat this process until all the nodes are localized.

Similar incremental localization methods can also be done by using angles [76, 77]. For these incremental methods, two issues need to be addressed. One is to deal with cascading error accumulation in large networks. One can adopt optimization techniques such as mass-spring relaxation to smooth out the error distribution, or, use robust statistics to handle outliers in input measurements [66]. The other issue is to handle insufficient number of initial anchor nodes. If there are a few anchors or the anchors are not well distributed, some nodes may not be able to find three neighboring anchor nodes to localize themselves. In this case, one can use distance estimations to anchor nodes via multi-hop paths [75] or adopt collaborative multi-lateration by solving a larger optimization problem [86, 87].

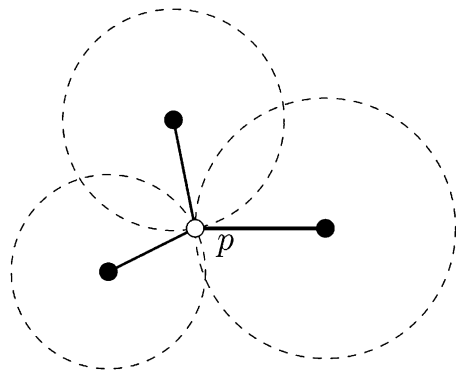


Fig. 5.1 A node p obtains its location by trilateration with distance measurements to three anchor nodes

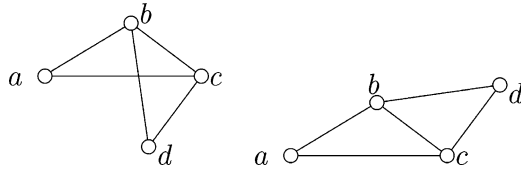


Fig. 5.2 A connectivity graph with two distinct embeddings having the same set of edge lengths

Existing anchor-free algorithms take either the connectivity graph [89] or the distances between neighboring sensor nodes [7, 42, 71, 91] as input. One major challenge with this approach is localization ambiguity – when the localization solution is not unique, localization algorithms may come up with a different embedding that satisfies all the distance constraints but deviates far from the ground truth. See Fig. 5.2 for an example. Indeed, with range information local optimization such as mass-spring relaxation techniques may get stuck at local minima with part of the network flip over the rest and generate a network layout far away from the ground truth [24, 71].

To deal with localization ambiguity, Moore et al. [71] proposed to use robust quadrilaterals as the basic iterative operation. A quadrilateral on four nodes with all pairs of edges is a globally rigid¹ component with a unique realization. The global layout is obtained by gluing locally identified robust quadrilaterals. Similarly, with ideas from rigidity theory, Goldenberg et al. proposed to record, propagate, and verify multiple possible locations of sensors to discover the truth network layout [42].

Global optimization techniques for sensor network localization include multi-dimensional scaling (MDS) [9, 89] and semidefinite programming [7, 91]. They formulate the problem as solving a global optimization problem for the sensor locations such that the distance constraints are satisfied. The results are typically better than local optimization algorithms. Similarly, with angle measurements one can formulate a linear program that solves for the node locations [13]. But these algorithms are centralized solutions. The performances of different localization algorithms have been evaluated in a Mote-based testbed [104, 105].

From a theoretical point of view, one can formulate the localization problem as embedding a unit disk graph in the plane if we assume the communication graph follows the unit disk graph model.² With purely connectivity information, determining whether a combinatorial graph is a unit-disk graph is NP-complete, and thus finding such an embedding in the plane (with neighboring nodes embedded within distance 1 and nonneighboring nodes more than distance 1 away) is also NP-hard [12]. In fact, even a relaxed version of the problem is still hard. Kuhn et al. proved that finding an embedding such that nonneighboring pairs are at least 1 away

¹ A graph is *rigid* in the plane is one can not continuous deform the shape of the graph without altering the lengths of the edges. A graph is *globally rigid* if it admits a unique embedding in the plane, subject to global rotation and translations.

² Two nodes within distance 1 are connected by an edge in a unit disk graph.

and neighboring pairs are within $\sqrt{3/2}$ is NP-hard [60]. Even with distance measurements of neighboring nodes or angle measurements of neighboring edges alone the problem is still NP-hard [3, 5, 13]. Not much is known on approximation algorithms for unit disk graph embedding. So far the only known theoretical result is an algorithm with an upper bound $O(\log^{2.5} n \sqrt{\log \log n})$ on the ratio of the longest distance between neighboring pairs to the shortest distance between nonneighboring pairs [60]. However, this algorithm uses some heavy graph embedding machinery and is not practical.

In practice, there have been a number of systems designed for indoor and outdoor localization. Most of them assume some anchors or beacon nodes with known locations, such as WiFi access points. One prevailing strategy is to use RF fingerprinting a priori to collect signal strength values from different access points tagged with location information. Then a node can be localized by matching the current signal strength values it receives from different access points with the RF fingerprints. A few representative systems include RADAR [6, 58, 63, 107], Intel's Place lab system [20]. Another category of indoor localization uses directional antennas and angle of arrival (AoA) to localize a node. In VORBA [78], the anchor nodes are equipped with rotating directional antennas to estimate AoA information from packets transmitted from a node to be localized. A simple triangulation approach is used to find its position. Approaches for indoor localization using other mediums such as ultra sounds (Active Bat [48]), infrared (Active Badge [102]), optical waves [72] have also been proposed. In this category, a widely known system is the Cricket [80] system from MIT. It uses ultrasound combined with RF. It uses several beacons that transmit ultrasound waves deployed on the ceiling of each room in the building. The nodes receiving these waves infer the range and localize themselves.

5.3 Geographical Routing

Geographical routing was originally proposed by Bose et al. [10], and independently by Karp and Kung [53] for routing in resource-constrained and dynamic networks such as ad hoc mobile networks and sensor networks. Further improvement on worst-case efficiency was proposed in the GOAFR+ family protocols [61, 62]. In geographical routing, the physical locations of the sensor nodes are used to guide information routing. Geographical routing has two modes: greedy mode and recovery mode.

5.3.1 Greedy Mode

In this mode, the node currently holding the packet “advances” it toward the destination, based only on the location of itself, its immediate neighbors, and the destination. The advance to the destination may be defined in many ways. Examples are, closest to destination [10, 30, 53, 96], most forward within radius (MFR) [96],

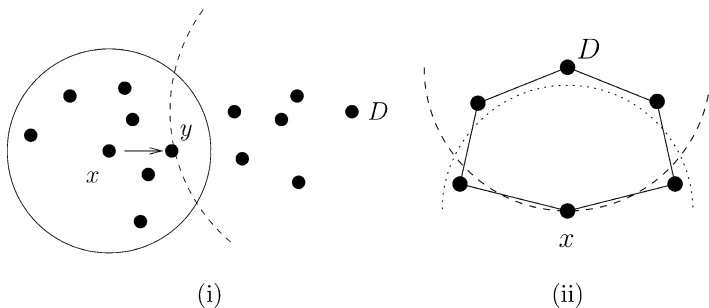


Fig. 5.3 (i) In the greedy forwarding mode, x sends the packets to y , the neighbor closest to the destination D ; (ii) x is a local minimum whose neighbors are all further away from the destination D than x itself

nearest forward progress (NFP), nearest closer (NC) [95], geographic distance routing (GEDIR) [94], and compass routing [57]. The most popular way of defining the advance is to examine the Euclidean distance to the destination and select the next hop as the node closest to the destination, as shown in Fig. 5.3. The greedy routing often suffices to deliver a packet in a dense network, but may fail in a sparse network. For example, the packet may reach a node called a *local minimum* whose neighbors are all further away from the destination than itself, as shown in Fig. 5.3.

5.3.2 Recovery Mode

When the greedy mode fails to advance a packet at some node, the routing process is converted to the recovery mode. The recovery mode defines how to forward the packet at a local minimum, to guarantee delivery of the packets. Some examples of the methods to get out of the local minimum are simple flooding [94], terminode routing [8], bread-first search or depth-first search [51], face routing algorithm [10], and perimeter routing [53].

Here we use a specific routing protocol, the greedy perimeter stateless routing (GPSR) protocol [53], to explain the two modes in detail. In the greedy mode, a node forwards the packet to its neighbor who is closest to the destination. The packet may reach a *local minimum* whose neighbors are all further away from the destination than itself. To get out of the local minimum, the protocol maintains a planar and connected subgraph, e.g., *Gabriel graph* (GG) or *relative neighborhood graph* (RNG) (see Fig. 5.4).

When a packet gets stuck, the recovery mode applies routing along the faces of the subgraph that intersect the imaginary line between the source and the destination (Fig. 5.5). When the message reaches a node closer to the destination than the node where it enters the recovery mode, greedy routing is adopted again. Greedy routing combined with the perimeter routing guarantees the delivery of a packet to the destination if there is a path. This recovery mode has been independently discovered by Bose as the face routing algorithm [10]. The construction of the planar

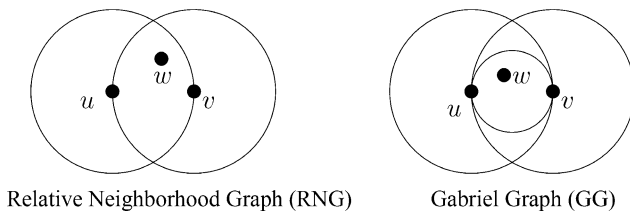


Fig. 5.4 In RNG, the edge uv is included unless there is a node w in the intersection of the two disks centered at u, v with radii equivalent to the Euclidean distance between u and v . In GG, the edge uv is included unless there is a node w in the disk with u, v as diameter. Both RNG and GG are planar subgraphs of the communication graph, modeled as a unit disk graph

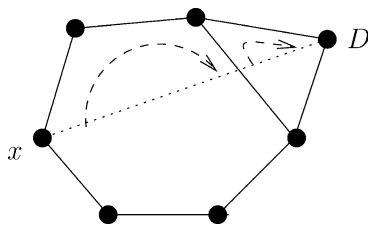


Fig. 5.5 Perimeter routing along the faces of a planar graph from x to the destination

graphs as well as the routing decisions requires localized decisions with only the information of neighboring nodes. This makes geographical routing very attractive for large-scale networks as the state information maintained in each node and in the packet is nearly minimal.

The recovery scheme with face routing or perimeter routing works nicely in theory but encounters a few problems in practice. The correct construction of a planar subgraph assumes accurate location information that is hard to obtain, and that the communication graph is modeled by a unit disk graph, which is not true in practice. Experimental evaluations of the communication model on sensor nodes reveal various spatial and temporal radio irregularities [38, 110]. Close-by nodes may not be able to communicate directly while some long and high-quality links may exist. Over time link quality also varies. In practice, the relative neighborhood graph or Gabriel graph extracted may become disconnected or still contain crossing edges, as shown in [55, 88]. This subsequently causes the delivery rate drop to about 68% on a real testbed [55]. Later, a number of repair mechanisms propose to remove crossing links by probing techniques [45, 54, 55]. These mechanisms improve the delivery rate with extra processing.

Another problem of face routing, especially in a sensor field with big holes, is that the routes are likely to be “hugging” the hole boundaries. Thus the nodes on the hole boundaries experience higher traffic than the average. The load imbalance may cause the boundary nodes run out of battery earlier than the others, enlarging the size of the holes and/or disconnecting the network.

To summarize, the challenge of localization and extracting a planar subgraph and the problem of traffic accumulation on the hole boundaries are the two major

motivations to move beyond geographical routing and design virtual coordinates that still retain the good properties of geographical routing, including the localized routing algorithm, near minimum state information, and robustness to topological changes and node failures.

5.4 Routing with Virtual Coordinates

5.4.1 Virtual Coordinates

5.4.1.1 NoGeo

NoGeo [81], by Rao et al. is one of the earliest proposals for efficient geometric routing with virtual coordinates. When sensor locations are not available, the idea is to generate virtual coordinates for the sensor nodes and apply standard geographical routing on these virtual coordinates. The virtual coordinates are defined with respect to the network connectivity, in particular, are produced by the rubberband representation of a graph [67]. To get a rubberband representation, some nodes (preferably on the network perimeter) are given fixed locations, each node (that is not fixed) runs an iterative algorithm to put itself at the center of mass of the neighbors' current locations. At each iteration, a node's location is updated as the average of the neighbors' current locations:

$$x_i = \frac{\sum_{j \in N(i)} x_j}{n_i}; \quad y_i = \frac{\sum_{j \in N(i)} y_j}{n_i},$$

where (x_i, y_i) is the current location of node i , $N(i)$ is the set of neighbors of node i and $n_i = |N(i)|$. This algorithm is known to reduce the total energy of the system, if all the edges are considered as rubberbands, and converge to a unique drawing what is called the rubberband representation.

When the nodes on the perimeter are identified but their locations are unavailable, a bootstrap stage is introduced to first embed the nodes on the perimeter. Each node on the perimeter floods the network and every other node on the perimeter records the minimum hop count distance to all other nodes on the perimeter. Now each node on the perimeter runs the following optimization to find an embedding that minimizes the sum of squared difference between the embedded pairwise distances and the hop count measurements:

$$\min \sum_{i,j \in P} [h(i, j) - d(i, j)]^2,$$

where P is the set of nodes on the perimeter, $h(i, j)$ is the hop count between two perimeter nodes i, j and $d(i, j)$ is the distance of i, j in the embedding to be found.

This optimization can be solved by running a standard multidimensional scaling algorithm [9] at each perimeter node.

Last, if the nodes on the perimeter are not known, we can identify them in the following manner. A couple of nodes, chosen arbitrarily, are designated as the bootstrapping beacons. They flood the network and the nodes with maximum hop count values within 2-hop neighborhood are identified as perimeter nodes.

As the virtual coordinates have no reason to be close to the real locations, only the greedy algorithm is applied for routing on these virtual coordinates. When a packet gets stuck at a node u , *scoped flooding* is adopted to deliver the packet. In particular, u performs a restricted flooding scheme and floods the network with TTL initially set to 1. Only the nodes within TTL-hops from u receive the message. TTL is then doubled each time until the destination is reached.

The performance of greedy routing on the constructed virtual coordinates is shown in [81] on a number of topologies. Surprisingly, the greedy forwarding scheme is quite robust and achieves a high delivery rate given sufficient node density. When the network has big holes due to obstacles, the performance of greedy routing on virtual coordinates can be even better than the performance of routing on true locations. Intuitively, this is because the virtual coordinates, derived from the network connectivity, reflect the network connectivity better than the true geographical locations.

5.4.1.2 Greedy Embeddings

A few theoretical work motivated by NoGeo asks whether one can find an embedding of a given graph in the plane such that greedy routing is guaranteed to work [19, 79]. Such an embedding is called a *greedy embedding*. In other words, for every pair of nodes p, q , there is a neighbor of p whose distance to q is smaller than the distance from p to q . It is known that not every graph admits a greedy embedding, such as a star with 7 leaf nodes [79]. Some graphs are known to have a greedy embedding, for example, graphs with a Hamiltonian path, any complete graphs, any 4-connected planar graphs (as they have Hamiltonian paths [97]), and any Delaunay triangulations. It still remains open to fully characterize the class of graphs that admit a greedy embedding.

Papadimitriou and Ratajczak [79] made the following conjecture that any planar 3-connected graph³ has a greedy embedding in the plane. They show that a planar 3-connected graph has an embedding in 3D such that with a special distance function greedy routing always works. This is due to a famous graph theory result that a 3-connected planar graph is actually the edge graph of a 3D convex polytope [111]. Thus one can find a distance function to route along the surface of this convex polytope that guarantees delivery. Raghavan Dhandapani discovered that any planar triangulation admits a greedy embedding in the plane for which greedy forwarding always succeeds [22].

³ A graph is 3-connected if it remains connected after the removal of any 2 nodes.

Recently the 3-connected graph conjecture was proved to be true by Leighton and Moitra [64], and independently by Angelini et al. [2]. Later the algorithm in [64] was improved such that the coordinates use $O(\log n)$ bits for a graph with n vertices [43].

A recent observation by Kleinberg [56] shows that if we use hyperbolic space then greedy routing becomes easy. He showed that any connected graph has an embedding in the hyperbolic space such that by using the hyperbolic distance greedy routing from any node to any node always succeeds. The intuition is to embed a tree in a hyperbolic space such that greedy routing works on the tree. Since any connected graph has a spanning tree, greedy routing works for all connected graphs. This observation also leads to a distributed algorithm to assign virtual coordinates to the sensor nodes in a hyperbolic space.

5.4.1.3 Virtual Polar Coordinate Routing (GEM)

Virtual polar-coordinate routing (GEM), proposed by Newsome and Song [73], constructs a *polar* coordinate system as the virtual coordinates for the sensor nodes. Specifically, take a spanning tree on the sensor nodes and give each node a polar coordinate with respect to its position in the tree. A node's virtual coordinate includes two components: its *level*, i.e., minimum hop count to the root of the tree, and its *angle range*. The root of the tree is given the largest angle range, e.g., from 0 to $2^{16} - 1$. The child of a node p is given a subset of the range of p , proportional to the size of its subtree. The angle ranges of two children do not overlap (Fig. 5.6).

With this polar coordinate system the following greedy routing scheme, named the virtual polar coordinate routing (VPCR), is used to deliver messages from one node to another. In VPCR, a node checks its 1-hop neighbors to look for any node that is “closer” to the destination. The “closeness” or distance of two angle ranges is defined as the minimum distance between two angles in the respective ranges. If a node does not find any neighbor closer to the destination, then the packet is routed to the parent. Notice that when the packet reaches the root of the tree, the packet is delivered down the tree structure to the child whose subtree contains the destination. Therefore, the algorithm eventually delivers the message. Similar with NoGeo [81],

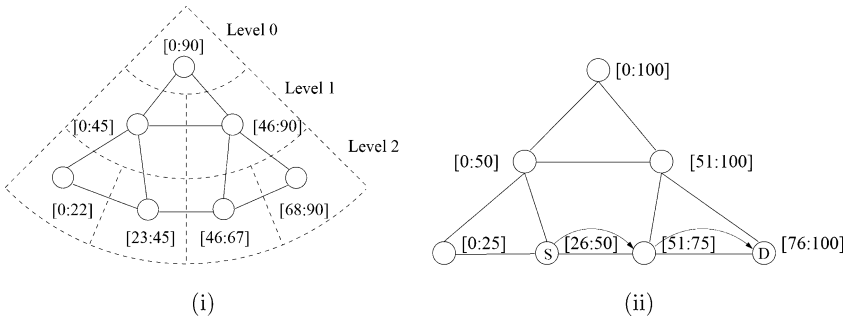


Fig. 5.6 (i) The polar coordinate assignment; (ii) Virtual polar coordinate routing

GEM also maintains the constant storage requirement and $O(1)$ state information in the packets, as well as the localized routing steps.

One issue with the polar coordinates is that traffic tends to accumulate around the root of the tree and its neighborhood. To alleviate this, multiple trees can be constructed and a packet is routed randomly on one tree and the set of polar coordinates associated with it.

5.4.1.4 Face Tracing

Motivated by the challenges of extracting a planar subgraph from the communication network for face routing with geographical locations, Zhang et al. proposed to define a “face” in a generic way and conduct face routing on an arbitrary graph [108]. At each node v , a fixed and arbitrary cyclic ordering of the adjacent edges is defined, and named as a “rotation”. Each edge of v is given an index as an integer from 0 to $d - 1$, where d is the degree of v . The edge vw with index $i + 1$ is said to follow the edge vu with index i . The edge with index 0 follows the edge with index $d - 1$. An edge uv has two directions, one is from u to v and one is from v to u . With a fixed rotation scheme on the vertices, one defines a “face” by traversing the edges following the rotation scheme. For example, start with a directed edge uv , the adjacent edge on the face is determined as the next edge at v following the rotation scheme at v . See Fig. 5.7. The face tracing algorithm visits v from the directed edge uv and takes the next edge vw by the cyclic index ordering at v .

The face tracing algorithm traces a sequence of directed edges by following the orderings of the rotation schemes at the vertices involved, and stops when the first directed edge is encountered again. Note that the first directed edge that is encountered the second time by the face tracing algorithm must be the first edge on the sequence, i.e., no directed edge can appear twice on a face boundary – if a different directed edge say vw is encountered the second time during face tracing, then the preceding edge uv must have been encountered twice as well, as the only way to take the edge vw is to come from the edge uv unless vw is the first edge on the sequence. Thus each face traced by the algorithm is a closed cycle of directed edges.

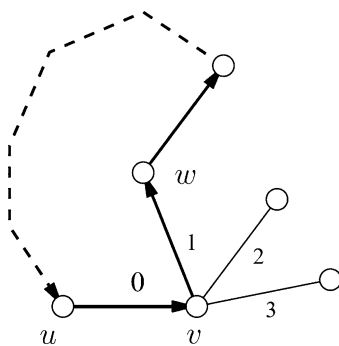


Fig. 5.7 Following the rotation scheme at each vertex, traversing a directed edge finds a face

Notice that the face tracing algorithm requires nothing beyond a fixed but arbitrary ordering of the edges at each vertex. Thus it works fine on a general graph and does not assume unit disk graph model. The face tracing algorithm is performed as a preprocessing phase to discover all the faces adjacent to each node. Each face is given a face ID. Now, routing with geographical coordinates consists of the standard greedy routing mode and the *face tracing* mode. When a message gets stuck at a node u , unable to make progress towards the destination, it enters the face tracing mode and traverses the graph by following a face adjacent to u . If the message discovers a node closer to the destination than u , it leaves the face tracing mode and returns to greedy routing mode. Otherwise, the message is routed to an adjacent face, that is, a face that shares an edge with the current face. The message saves the IDs of the faces it has visited, and hops from face to face until either it reaches the destination or traverses the whole graph and declares there is no path to the destination.

Different from the approach of using probing to remove crossing edges in an unsuccessful planar graph subtraction [45, 54, 55], the face tracing technique does not remove any links in face routing mode.

5.4.2 Landmark Routing

A number of virtual coordinate routing systems are landmark-based [16, 26, 32, 74]. The idea of using landmarks for generating node names or addresses and routing in large networks appeared two decades ago, such as the landmark hierarchy by Tsuchiya [100]. We will discuss hierarchical routing in Sect. 5.6. All the landmark-based routing schemes described below have a two-level hierarchy.

In sensor network setting, a small subset of the nodes is selected as *landmarks*. The landmarks flood the network and every node records its hop count distances to these landmarks. The landmark distances are then used to generate virtual coordinates for routing between the nodes. Routing is typically guided in a greedy way by a potential function on the landmark-based distances. The major differences among the following schemes are mostly in the design of the potential functions, as will be explained later.

Landmark-based schemes are favored by their simplicity and independence on the dimensionality of the network. Unlike face routing in planar graphs that requires a planar deployment of sensor nodes, landmark-based routing schemes can be easily extended to sensors deployed in 3D.

5.4.2.1 Gradient Landmark-Based Routing (glider)

Gradient landmark-based routing (glider) [26] is concerned with routing in a sensor network with a few big holes or an irregular shape. It separates global and local

routing by encoding the global topology of the network with a compact combinatorial graph, which provides a rough global routing guidance. The actual route with the global guidance will be realized with a greedy routing rule based on the landmark distances.

In glider, a number of landmarks are selected and the network is partitioned into *Voronoi tiles* such that all the nodes inside the same tile are closest (in terms of hop count distance) to the same landmark. Two Voronoi tiles are adjacent if there are two neighboring nodes in different Voronoi tiles. The tile adjacency graph, denoted by the *combinatorial Delaunay graph*, is abstracted and disseminated to all the nodes, for global routing across tiles. Figure 5.8 shows the landmark Voronoi diagram and the dual combinatorial Delaunay graph.

Each node p is given a virtual coordinate for routing. Once the Voronoi tiles are partitioned, each node belongs to a Voronoi cell. We call that cell the *resident tile* of the node, and we call its landmark the *home landmark*. Nodes with equal distance to multiple landmarks will choose the landmark with smallest ID as the home landmark. The neighbors of p 's home landmark on the combinatorial Delaunay graph are called the *reference landmarks*. The *name* of a node includes the ID of its home landmark, and the list of hop count distances to its reference landmarks. Notice that the name of a node is locally defined by the nearby subset of landmarks.

To route to a node in a different tile, a node first consults with the combinatorial Delaunay graph to find a sequence of tiles to visit. The actual routing across tiles is called *inter-tile routing* and can be implemented with the virtual coordinates of a node itself and its neighbors. Recall that every node has the minimum hop count

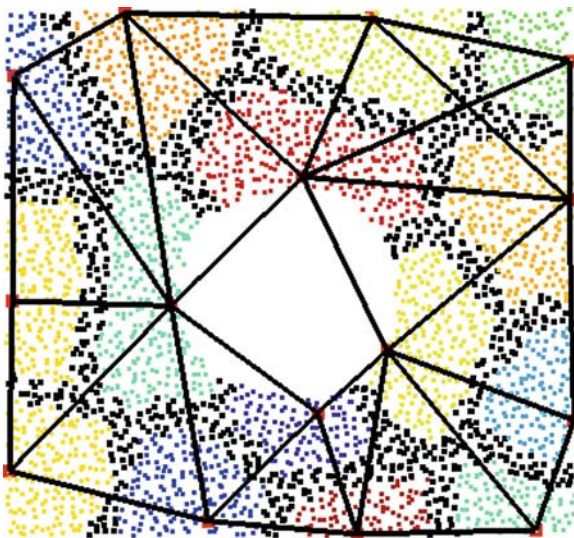


Fig. 5.8 Landmarks are shown by *large squares*. Sensor nodes are shown as *small dots*. The nodes are divided into tiles. All the nodes in the same tile are drawn in the same color. The dark nodes are the boundaries of the tiles

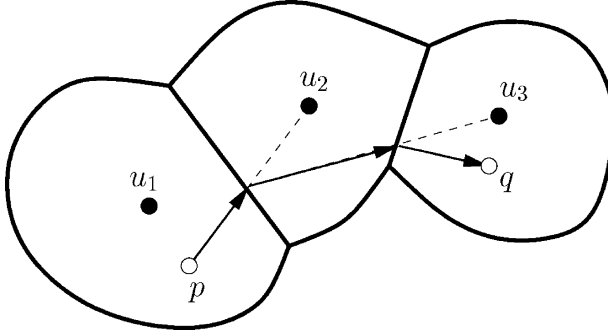


Fig. 5.9 Routing across tiles: to route toward the tile of u_2 the packet is delivered in a greedy fashion towards the adjacent landmark u_2 until it enters the tile

information to its reference landmarks. Thus a packet destined to an adjacent tile is routed toward the landmark in that tile, in particular, forwarded to the neighbor whose hop count distance to the landmark of that tile is reduced. See Fig. 5.9. When a packet enters the next tile, it checks whether this tile contains the destination. If not, inter-tile routing is used again to route to the next transit tile. We remark that such transition happens when the packet first enters a tile, and thus the packet does not necessarily go through the landmarks in the transit tiles.

To route to a node in the same tile, the packet is guided by greedy descending on a potential function constructed with distances to a set of local landmarks (including the home landmark and the reference landmarks), denoted by $\{u_1, u_2, \dots, u_k\}$. For any node p , let $\tau(p, u_i)$ denote the minimum hop count distance between p and u_i .

Let $\bar{\tau}(p) = \sum_{i=1}^k \tau(p, u_i)^2 / k$ as the mean of the squared distances. We then assign to p the *centered virtual coordinate vector*

$$C(p) = (\tau(p, u_1)^2 - \bar{\tau}(p), \dots, \tau(p, u_k)^2 - \bar{\tau}(p)).$$

The centered virtual distance between two points p and q is then defined as $d(p, q) = |C(p) - C(q)|^2$. Given a destination q , our greedy routing algorithm chooses the neighbor r of p , which minimizes $d(r, q)$. That is, we move packets by greedy minimization of the Euclidean distance to the destination, measured in the virtual coordinate system. This algorithm is local and efficient since only the virtual coordinates of the neighbor nodes are needed.

The selection of the centered virtual coordinate vector is motivated by the fact that the variant in the continuous domain is free of local minimum as long as there are at least three local landmarks that are not colinear. The subtraction of the mean from the squared distance vector in the centered virtual coordinate is important. As shown in Fig. 5.10 if we do not subtract the mean (i.e., define the virtual coordinate vector as the squared distance vector) routing may get stuck at local minimum.

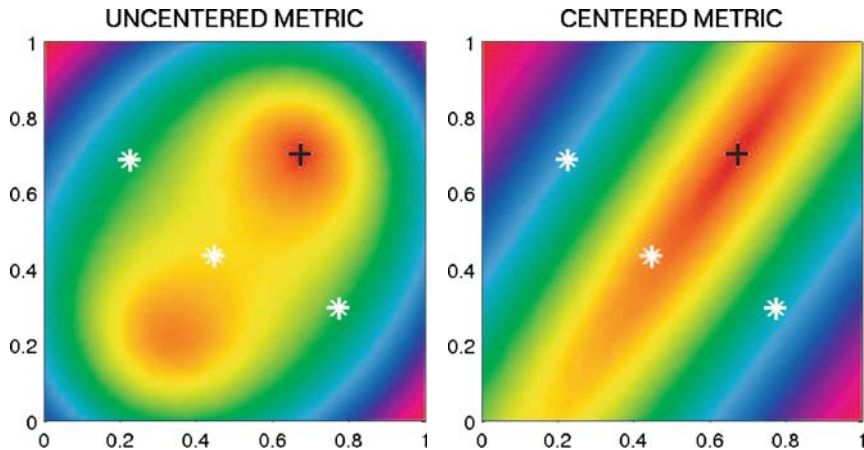


Fig. 5.10 The distance function for landmark-based greedy routing. There are three landmarks marked by *snowflakes*. The destination is marked by a $+$ sign. The color of a point represents its distance to the destination with respect to uncentered coordinates and centered coordinates, respectively. Note the local minimum in the uncentered case

In a discrete network inter-tile routing is always successful but intra-tile routing with the centered virtual coordinate vector may still get stuck, if the sensors are distributed sparsely. Experiments show that when the randomly distributed nodes have on average about 6 or more neighbors the delivery rate is close to 100%. In the worst case when a packet arrives at a local minimum, the node can simply flood the tile containing the destination to deliver the message.

To summarize, glider separates the global routing (how to get around holes) and local routing (to deliver a packet to the destination) by the two-level hierarchy. In many of the real-world situations where sensor networks may be deployed, the topological features of the layout (e.g., holes) will be only a few and will mostly reflect the underlying structure of the environment (e.g., obstacles). Moreover, this relatively simple global topology is likely to remain stable: nodes may come and go, but such spontaneous changes are unlikely to destroy or create large-scale topological features. It follows, since the global topology is stable, that we can afford to carry out proactive routing at the abstract combinatorial Delaunay graph level and realize the high-level routes as actual paths in the network by using reactive protocols by greedy routing with the virtual landmark-based coordinates.

The design principle of glider hinted that the landmarks should be selected to capture the global topology of the sensor field. Thus the number of landmarks is likely to be dependent on the topological complexity, such as the number of holes in the sensor field, rather than the number of sensors. A very recent work [40] investigated this problem in a continuous domain and established criteria on how the landmarks are placed to guarantee that the combinatorial Delaunay graph (or its continuous variant, the geodesic Delaunay triangulation) represents the exact topology of the underlying (unknown) domain. In companion, a distributed algorithm on how the landmarks are selected in a sensor network setting is also proposed in [40].

5.4.2.2 Beacon Vector Routing (BVR)

Beacon vector routing (BVR) [32] is another landmark-based routing scheme that has received significant attention. It uses a potential function that depends on the distances to the k landmarks closest to the destination, where k is a system parameter. Out of the k landmarks, the ones that are closer to the destination than to the source impose a “pulling” force while the rest impose a “pushing” force. The potential function is a combination of the two.

For the destination node q , let $\tau(p, u_i)$ denote the minimal hop count between q and a landmark u_i . Denote by $C_k(q)$ as q ’s k closest landmarks. Denote by $C_k^+(p, q) \subseteq C_k(q)$ as the subset of landmarks that are closer to q than p ; and $C_k^-(p, q) \subseteq C_k(q)$ as the subset of landmarks that are further away from q than p . The potential function $\delta(p, q)$ is defined as

$$\delta_k(p, q) = A \sum_{i \in C_k^+(p, q)} [\tau(p, u_i) - \tau(q, u_i)] + \sum_{i \in C_k^-(p, q)} [\tau(q, u_i) - \tau(p, u_i)],$$

where A is a parameter taken as 10. The greedy routing chooses a neighbor that minimizes the above potential function to the destination. When the greedy routing gets stuck, the message is delivered to the landmark closest to the destination, from where a restricted flooding with increasing scope is performed to deliver the packet.

Although the potential function design is a heuristic, the algorithm works well as evaluated in simulations. BVR has been selected in a number of cases as a comparison benchmark [70, 74].

5.4.2.3 Greedy Landmark Descent Routing (GLDR)

Greedy landmark descent routing is recently proposed by Nguyen et al. [74], which, combined with a landmark selection strategy, guarantees packet delivery with bounded stretch in the continuous domain.

The landmarks are selected to be an r -sampling: any node is within hop count r from a landmark. These landmarks are chosen by a distributed algorithm running in parallel on all the nodes. Each node waits for a random period of time and declares itself to be a landmark, if it has not been suppressed by other landmarks. Once it becomes a landmark it notifies its r -hop neighbors and suppresses all nonlandmark nodes. This generates an r -sampling and depending on the parameters of the random waiting time the number of landmarks selected can be controlled.

Similar to BVR, a node’s virtual coordinate is defined as the distance vector to a small set of nearest landmarks called the *addressing landmarks*. To route a packet to its destination, the source node chooses, among the destination’s addressing landmarks, the one that *maximizes* the ratio of the distances to source and destination, and moves toward this landmark until it reaches a node with the same hop count to this landmark as the destination. At this point, the routing procedure is repeated and a different landmark is chosen until the destination is reached. An example is shown

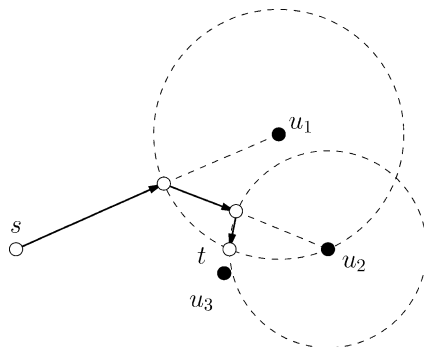


Fig. 5.11 Greedy landmark descent routing. The source s chooses the landmark u_1 and routes toward it until the distance to u_1 equals the distance from the destination t to u_1 . The same procedure is repeated until the destination t is reached

in Fig. 5.11. This scheme is known to always work in the continuous domain and generates a path with constant bounded stretch. In a discrete network it may happen that the packet gets in a loop, which can be detected by checking the last few landmarks toward which the packet moves. When this happens the packet is greedily forwarded to the destination using L_1 and L_∞ norms on the distance vectors of the addressing landmarks. If the destination is still not reached, a scoped flooding is used to deliver the packet.

The landmark selection scheme by using r -samplings is of independent interest and can be useful for other landmark-based routing schemes. For example, when the landmarks are selected as an r -sampling rather than a random sampling, BVR has improved delivery rate [74].

5.4.2.4 Macroscopic Geographic Greedy Routing

Funke and Milosavljevic proposed Macroscopic Geographic Greedy Routing that combines the idea of glider and geographical routing to account for inaccurate location information and small network holes and irregularities [36, 37]. In glider, a sparse set of landmarks is selected to capture the big topological features of the sensor field; the combinatorial Delaunay graph on the landmarks is distributed to every node for global routing guidance. In Macroscopic Geographic Greedy Routing, a dense set of landmarks is selected and the combinatorial Delaunay graph is not distributed to all the nodes but rather used for geographical routing.

The idea is to extract a planar subgraph of the combinatorial Delaunay graph and embed it in the plane. Each node v is given a name consisting of the home landmark, i.e., the landmark whose Voronoi tile contains v , and its unique ID. The intratile and intertile routing are implemented in the following way. For intratile routing, since the landmarks are dense and each tile has only a constant number of nodes,

the packet is simply flooded to all the nodes in the destination tile. For intertile routing, the packet is guided to the next tile whose home landmark is closer to the destination in terms of Euclidean distance. In other words, the next landmark is chosen by geographical greedy forwarding and the intertile routing in glider is used to route a packet to that adjacent tile. When the current home landmark does not have a neighboring landmark closer to the destination, perimeter routing is adopted to route on the planarized combinatorial Delaunay graph.

To summarize, the planarized combinatorial Delaunay graph and its embedding are used to navigate in the sensor field. This approach can be considered as in between glider and geographical routing. As the landmarks are reasonably separated, the navigation is robust to low-level network link variations and location inaccuracies.

5.4.2.5 Small State Small Stretch Routing

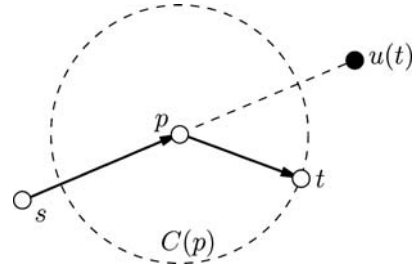
Mao et al. [70] adopted the idea of compact routing schemes (by Thorup and Zwick [98, 99]) for routing in sensor networks. The basic idea is to select about $O(\sqrt{n})$ landmarks. These landmarks flood the network and every node records the hop count distances to these landmarks. Denote by $u(p)$ the landmark that is closest to p and $r(p)$ the distance from p to $u(p)$. Each node p identifies its *cluster* $C(p)$, consisting of the nodes q whose distance to p is within distance $r(q)$, i.e., all the nodes that are closer to p than their closest landmarks. For every node p , it maintains a routing table with hop count values to all the landmarks and the nodes in its cluster $C(p)$.

The routing algorithm is a greedy algorithm with the following rules:

- If the destination t is inside the cluster $C(s)$ of the source s , then s routes to t directly with its routing table.
- If the destination t is outside the cluster $C(s)$ of the source s , then s first routes the packet toward the landmark closest to t . When it encounters a node with routing table entry to t , the packet is delivered to t .

One can see that this routing algorithm guarantees delivery – t is within the cluster of its closest landmark $u(t)$, thus the packet will eventually get to a node who has a routing table entry for t , from where the packet is guided by the local routing tables to the destination. In addition, the length of the routing path is within a factor of 3 of the minimum hop count distance between the source and the destination. Denote by $\tau(p, q)$ the hop count distance between p and q . If the destination is within $C(s)$, the packet is delivered through the shortest path with the help of the routing tables. If the destination is outside $C(s)$, as shown in Fig. 5.12, $\tau(s, t) \geq \tau(t, u(t))$. Thus the packet is routed toward $u(t)$. Suppose on the way to $u(t)$ the packet arrives at a node p whose cluster contains the destination t . We have $\tau(s, p) \leq \tau(s, u(t))$, $\tau(p, t) \leq \tau(u(t), t)$. Now the length of the routing path is

Fig. 5.12 An example of compact routing



$$\begin{aligned}
 \tau(s, p) + \tau(p, t) &\leq \tau(s, u(t)) + \tau(u(t), t) \\
 &\leq [\tau(s, t) + \tau(t, u(t))] + \tau(u(t), t) \\
 &\leq \tau(s, t) + 2\tau(u(t), t) \\
 &\leq 3\tau(s, t).
 \end{aligned}$$

When the sensor nodes are uniformly deployed, one can place about $O(\sqrt{n})$ landmarks uniformly such that the size of the cluster for each node is roughly $O(\sqrt{n})$ as well. Thus the size of the routing table is in the order of $O(\sqrt{n})$, with which a worst-case stretch 3 path can be established for any pair of nodes.

To conclude this subsection, we surveyed a set of representative ideas in using landmarks to define virtual coordinates for greedy routing in a sensor network. A few other protocols with similar ideas to some of these were not covered in details here. For example, Caruso et al. [16] used three landmarks and a heuristic potential function for routing in a sensor network. Wattenhofer et al. [103] studied the placement of landmarks for routing in special graphs such as rings, grids, trees, butterfly graphs, and hypercubes. At last we remark that these landmarks are mainly used as points of reference, and do not serve the functionalities of gateway nodes (which tends to attract network traffic). After the establishment of the virtual coordinates the landmarks typically behave in the same way as the other nodes.

5.4.3 Medial Axis-Based Routing

The medial axis-based routing, proposed by Bruck, Gao, and Jiang [14], considers a setting similar to the setting in glider, in which the sensors are uniformly deployed in a geometric region with possible obstacles or irregular shape. Again the global topology of the sensor field is captured for global routing guidance, the difference is in the use of the medial axis of the underlying sensor field for this purpose.

The *medial axis* of a sensor network is identified as the set of nodes with at least two closest nodes on the network boundary. A point on the medial axis is called a medial vertex if it has three or more closest nodes on the network boundary. See Fig. 5.13 for an example. The medial axis is a “skeleton” of a region that captures all the topological features, such as how many holes there are and how they are

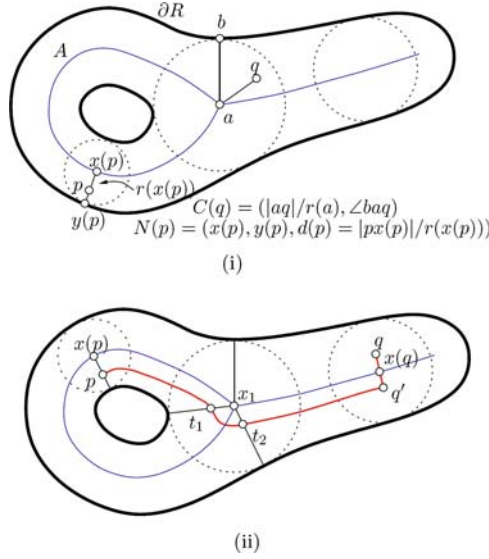


Fig. 5.13 An example of the medial axis of the boundary of a closed region R (only the part of the medial axis in the interior of R is shown). ∂R is shown by *thick curves*. The medial axis A has a cycle, which means that the region R has a punched hole. (i) The naming scheme. (ii) Routing from p to q by the road system

connected. For the purpose of routing, the medial axis is represented by a medial axis graph (MAG), which is a combinatorial graph connecting the medial vertices, and has a size proportional to the number of large geometric features. This medial axis graph is very compact and is known to every sensor. For example, the medial axis graph of Fig. 5.13 has two medial vertices, one edge and one self-loop. The construction of the medial axis requires the detection of nodes on the network boundaries (or hole boundaries), for which there exist distributed and efficient algorithms [25, 28, 29, 33, 35, 41, 59, 101]. The nodes on the medial axis can be identified through local flooding. Specifically, each boundary node initiates the flooding of a message, which contains its ID and a counter that records how many hops the message has traveled. If a node receives a packet from a boundary node that is further away from its current nearest boundary node(s), it stops forwarding this packet. If the boundary nodes initiate their flooding at approximately the same time, and each packet travels at approximately the same speed, then a packet is dropped once it reaches the medial axis. This approach cuts down the total number of packets delivered and keeps the total communication cost low. As a result, each node learns its nearest boundary node(s) and can determine whether it itself is on the medial axis. After the construction of the medial axis, we let a node flood the network, pull the information about the medial axis, and construct the abstracted medial axis graph. This graph is then broadcast to every sensor. In addition, each node on a medial edge remembers which medial edge it stays on, its neighboring nodes on that medial edge, and how many hops it is from each endpoint of the medial edge.

With the medial axis constructed, each node is given a name with respect to the medial axis graph, defined by the closest node on the medial axis and the hop count distance from it. For each sensor w on the medial axis, we define a chord as the shortest path (tree) from w to its closest sensor nodes on the boundary. A sensor's name includes the chord on which it stays, and a normalized distance to its corresponding medial axis node. See point p in Fig. 5.13. Such a naming scheme partitions the sensor field into canonical regions inside each of which a local Cartesian coordinate system is defined with one axis as an edge of the medial axis graph and the other axis as a chord of a vertex on that medial axis edge. The local Cartesian coordinate systems are glued together in exactly the same way as indicated by the edge adjacency of the medial axis graph, and provide a smooth and natural road system for efficient point-to-point routing. To help a packet route from one canonical piece to an adjacent one, we also establish a Polar coordinate system around each medial vertex, see node q in Fig. 5.13.

For a node to find a route to the destination, it only needs to know the medial axis graph and the name of the destination. Routing is first planned on the abstract medial axis graph, which is usually of a small size, and then realized in each canonical region by reactive local gradient descent routing. By using the medial axis graph in a global planning step, a source can find the *reference path*, defined as the shortest path in the medial axis graph, from the node in the medial axis corresponding to the source to the one corresponding to the destination. The actual routing rule is of Manhattan-type, i.e., first trying to match the medial axis point with that of the destination and routing in parallel with the reference path, and then trying to match the distance to the medial axis point with that of the destination and routing along chords. Both routing in parallel with the medial axis and along chords can be realized by efficient local gradient descending in the local coordinate systems of the canonical regions. See Fig. 5.13.

We show in Fig. 5.14 an example of a sensor network with a few large holes, the medial axis and the medial axis graph. Figure 5.15 shows the path generated by the medial axis-based routing and the path generated by GPSR. The medial-axis based routing scheme has accumulatively more balanced traffic load on the sensor nodes as routing paths are not converging to hole boundaries in such a complex network.

5.4.4 Ring-Based Routing Scheme

The geometric schemes we discussed so far use the fact that the sensors are typically deployed densely in a 2D region, to provide sufficient coverage and sensing resolution. A ring-based routing scheme, recently proposed by Caesar et al. [15], reduces the problem to routing in a 1D ring.

In virtual ring routing (VRR), the nodes are given arbitrary identifiers and organize themselves into a virtual ring in the cyclic increasing order of their IDs. Each node maintains r adjacent neighbors on the ring ($r/2$ neighbors clockwise and $r/2$ neighbors counter-clockwise). These neighbors are called *virtual neighbors* of a node, to be differentiated from its *physical neighbors* that are directly connected by

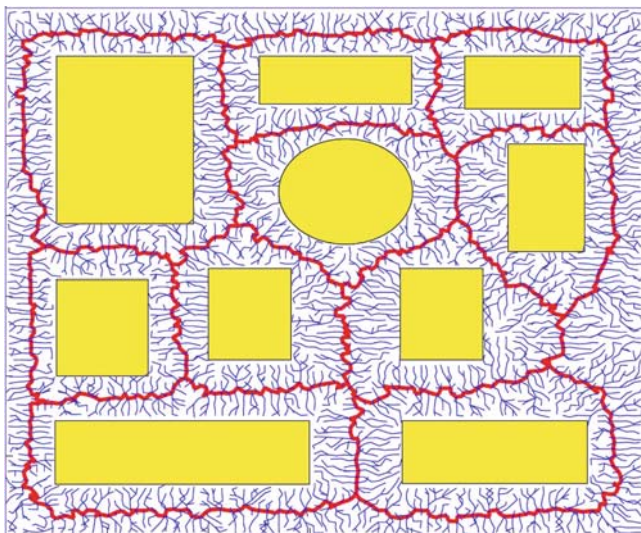


Fig. 5.14 The medial axis and the chords (shortest path tree rooted at nodes on the medial axis). The obstacles (holes) are colored in yellow

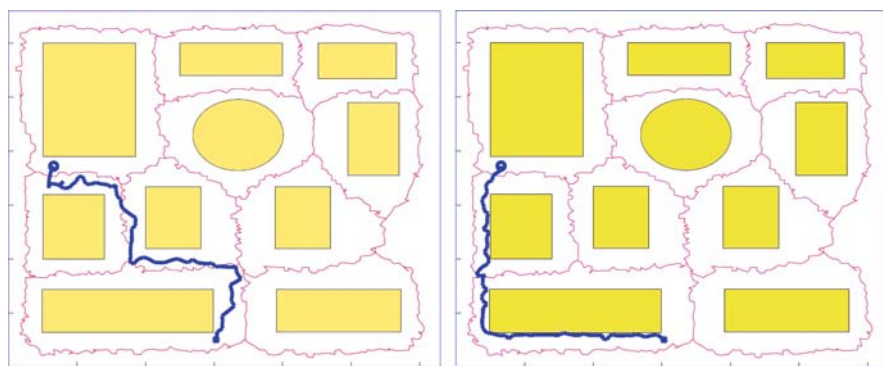


Fig. 5.15 The path generated by the medial axis based routing (*left*), in comparison with that of geographical routing (*right*)

wireless links. Notice that the virtual neighbors are by no means physically close. In virtual ring routing, the routes from a node to each of its virtual neighbors are established and stored in the routing tables of the nodes on the routes. See Fig. 5.16. These paths are called *vset-paths* and will be maintained when the network topology changes. Specifically, each node p maintains a routing table with the following entries $(S_i, T_i, \text{next hop toward } S_i, \text{next hop toward } T_i)$, for each pair of virtual neighbors (S_i, T_i) whose path goes through p .

When a packet is routed to the destination, if the source node has a routing table entry for the destination then the packet will be delivered accordingly. Otherwise,

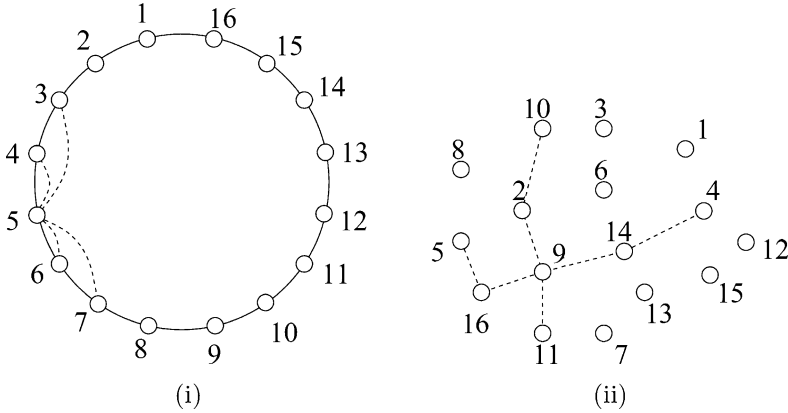


Fig. 5.16 (i) Virtual ring and the virtual neighbors of node 5. (ii) The actual paths between node 5 and its virtual neighbor 4, and node 10 and its virtual neighbor 11 in the real network. All the nodes on the paths maintain routing table entries for these paths

the source s checks its routing table and picks the path endpoint q closest to the destination ID on the virtual ring. The packet is forwarded to the next hop toward q . At the next node this operation is repeated. One can see that with the virtual paths stored in the network one can route from any node to any node – in the worst case the packet can follow the virtual ring to reach the destination. The actual performance can be better as each node also maintains routing information for the virtual paths that go through it. Intuitively, a node is routed toward the node closest to the destination in the ID space (on the virtual ring), although that node might be in a completely different direction from the destination. But the nodes close to the destination on the virtual ring have routing table entries to the destination. Thus the algorithm routes a packet to some nodes with more knowledge of the destination. In fact, it often happens that a packet does not have to visit the virtual neighbors of the destination before it finds routing information to the destination. Simulation results show that the routing path actually has on average a small stretch compared with the shortest paths.

During the network initialization, the virtual paths can be constructed incrementally without flooding the network. The control messages to build a new virtual path are routed with the current virtual paths. Any path failures caused by node mobility or topological changes are repaired in a similar way by using the remaining virtual paths, to adapt to network dynamics.

5.5 Information Discovery and Data-Centric Routing

In data-centric routing, a node poses a query for data of certain types, and the routing algorithm routes the query to retrieve the relevant data in the sensor network. Data-centric routing is fundamentally different from traditional routing paradigms as the

problem is not only concerned about getting a packet from a source node to the destination node, but also involves the discovery of the destination, i.e., the sensor nodes that have the required information. This can be formulated as the problem of *information brokerage* that specifies how data are collected, processed, and stored as well as how queries are routed to discover the relevant data. The problem is to match *information producers* (also known as sources), which perform data acquisition and event detection, with *information consumers* (also known as sinks) that search for this information.

This problem in sensor networks is initiated in the work, directed diffusion, by Intanagonwiwat, Govindan, and Estrin [50]. Sensor nodes locally process sensing data and organize it by attribute-value pairs. A node requests data by sending out *interests* for the named data. The interests are disseminated in the network. Once the data matching the requested interests is discovered, it is delivered back to the query node. The intermediate nodes may also cache interests and reinforce certain routes from data sources to the query node. A similar routing paradigm is also adopted in TinyDB [69] to support query of aggregated information of the distributed data. A SQL-style query is disseminated to all the nodes in the network during which a tree rooted at the query node is established. As the data are delivered back to the root, it is aggregated at the internal nodes. In this approach, little collaborative pre-processing is performed. Thus the discovery of the desired information usually relies on flooding the network. This approach targets at infrequent queries for streaming data type so that the cost of flooding can be justified and amortized by the following long-term data delivery.

5.5.1 Geographical Hash Tables

A parallel approach to data-centric routing is to adopt data-centric storage that targets at large-scale networks with many simultaneously detected events that are not necessarily desirable for all users [83, 90]. The idea is similar with Distributed Hash Tables (DHTs) on the Internet [82, 84, 92, 109]. A producer leaves data on rendezvous nodes for consumers to retrieve. Thus data across space and time can be aggregated at rendezvous nodes.

In geographical hash tables (GHTs) [83], data are hashed by data type to geographical locations. The nodes close to the hashed location serve as rendezvous nodes. The consumer applies the same hash function and retrieves data from the same set of rendezvous nodes. Data and query delivery to the rendezvous nodes is implemented by geographical routing such as GPSR [53]. Specifically, each piece of event, based on its data type, is hashed with a content-based hash function to a geographical location. This hashed location may not have a sensor node right there. The node *geographically closest* to the hashed location is considered as the *home node*. Then the event is routed with GPSR toward the hashed location. This packet will eventually reach the home node, by the property of geographical routing. At the home node, GPSR will not be able to find a neighbor closer to the hashed location

and thus enter the perimeter mode to tour the face on which the home node stays. This face is named the *home perimeter*. The nodes on the perimeter, other than the home node, are named the *replica nodes*. After the packet returns to the home node, the data caching component is accomplished. All the nodes on the home perimeter cached the data. To account for possible changes to the network topology, the home node periodically sends refresh packets to the hashed location to repair the home node and the home perimeter.

GHTs have greatly reduced the communication cost and energy consumption by avoiding network-wide flooding for information discovery. Its simplicity is also very appealing. This idea is further developed in [46] to build a hierarchical storage structure that is aware of data correlations, i.e., similar data are hashed close by.

5.5.2 Double Rulings

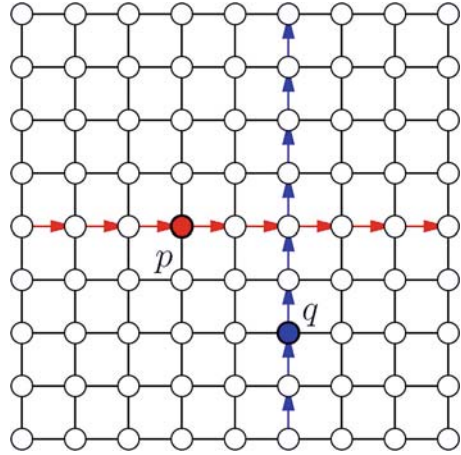
The idea of double rulings is an extension and improvement of GHTs, in terms of *distance sensitivity*. In particular, if the producer is actually close to the data producer (although neither has the knowledge of each other), we would like the data consumer to discover the data producer quickly. This is an attractive feature in many applications, as information will be most useful, thus queried more frequently, in the spatiotemporal locale where it was collected. In GHTs, the hashed location, typically generated by a random hash function uniformly on a sensor field, may be far away from both the producer and the consumer.

In double rulings, the rendezvous nodes are chosen along a continuous curve. The motivation is twofold. Data delivery from data source to a rendezvous node is implemented by multihop routing. Thus it is natural to leave information hints along the trail that the data travels on, at no extra communication cost. Furthermore, data hint replication on multiple nodes provides more flexibility for a consumer to discover relevant data – it is easier to encounter a 1D curve than a 0D node.

5.5.2.1 Rectilinear Double Rulings

A basic *double-ruling* scheme, developed in different variations for information discovery and routing [68, 93, 106], works as follows: data or data pointers are stored at nodes that follow a *replication curve* while a data request travels along a *retrieval curve*. Any retrieval curve intersects the replication curve for the desired data. Thus successful retrieval can be guaranteed. For an easy familiar case, assume the network is a two-dimensional grid embedded in the plane with nodes located at all the lattice points (see Fig. 5.17). The information storage curves follow the horizontal lines. The information retrieval curves follow the vertical lines. To be differentiated from the other double rulings schemes, we name this scheme the *rectilinear double rulings*. Notice that the data retrieval curves are independent of the locations of the data sources. In fact, a consumer traveling along the vertical line through itself

Fig. 5.17 The rectilinear double ruling scheme on a grid



is guaranteed to hit *all* horizontal storage lines, and thus is able to find *all* the data stored in the network. This double-ruling scheme is also distance sensitive – if the producer and consumer are actually near each other, they must also be near each other along the path connecting them using the horizontal and vertical lines.

5.5.2.2 Spherical Double Rulings

The spherical double rulings scheme, proposed by Sarkar, Zhu, and Gao [85], has both GHTs and the rectilinear double rulings as subcases. Same as in GHTs, a data item is hashed by its data type to a geographical location. However, instead of traveling along the geographical greedy path to the rendezvous node, the producer travels along a circle that goes through itself and the rendezvous node and replicates data or data pointers on the way.

For an easy explanation, we will use the stereographic projection to map sensor nodes onto a sphere [21]. We put a sphere with radius r tangent to the plane at the origin. Denote this tangent point as the south pole and its antipodal point as the north pole. A point h^* on the plane is mapped to the intersection of the line through h^* and the north pole with the sphere. See Fig. 5.18. This provides a one-to-one mapping of the plane to the sphere, in addition, with the north pole mapped to the point of infinity. Stereographic projection preserves circularity. Any circle on the sphere, including great circles, is mapped to a circle in the plane. With this mapping specified, the replication and retrieval schemes are described on a sphere.

Each data type is hashed to a geographical location h^* as in GHTs. When a producer routes toward the hashed location h^* as in GHTs, instead of following the geographical greedy route as in GHTs, it follows the great circle defined by its own location p and the hashed location h , denoted by $C(p, h)$. Data from different producers with the same data type will be routed to the same hashed location where information

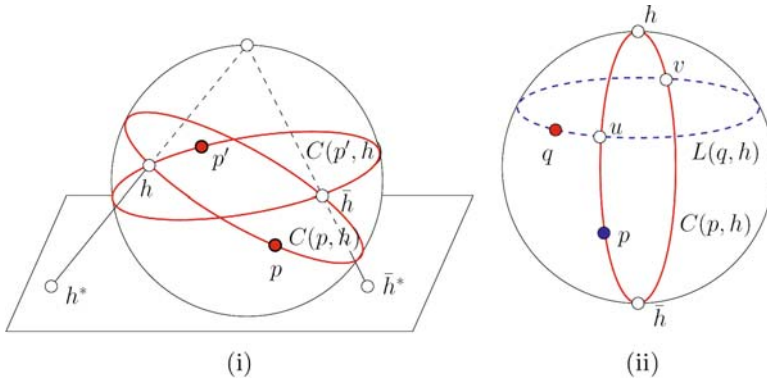


Fig. 5.18 (i) A point in the plane h^* is projected to a point h on the sphere. The great circles for two producers p, p' are drawn in red. (ii) The consumer follows the circle with fixed distance (dashed circle) to the hashed location h to retrieve all the data with the same data type

aggregation can be performed. All the great circles with type $C(*, h)$ pass through the hashed location h , as well as the antipodal point $\bar{h}$. Thus there are actually two rendezvous nodes, h and $\bar{h}$, located far away in the network that both have all the information of the same data type. Notice that the hashed location h depends only on the data type. Thus the location $\bar{h}$ can be derived by a simple geometric computation. See Fig. 5.18 for an example.

Now, this new data replication strategy from producers to hashed locations enables more flexible retrieval scheme for the consumer.

1. *GHTs retrieval rule*: the same as in GHTs, the consumer can route to the hashed location h^* or $\bar{h}^*$, whichever is closer, to retrieve all the data of the same type.
2. *Distance-sensitive retrieval rule*: If the consumer is of distance d from the producer, the consumer can discover the data with a cost of $O(d)$, although neither has the knowledge of each other's location or the bound on d . Specifically, if we rotate the sphere so that the hashed location h is at the north pole, then the replication curve is exactly a longitude curve. The distance-sensitive retrieval scheme follows the latitude curve searching for a replication curve. See Fig. 5.18 for an example. The retrieval curve (i.e., the latitude curve) has two intersections with the replication curve, one of which is within distance $O(d)$ from the consumer. Thus the consumer travels along the circle on the sphere with equal distance to the hashed location h , and uses a doubling trick⁴ to discover the closer intersection with the replication curve. An example is shown in Fig. 5.19.

⁴ The problem is to find a destination on a circle without knowing whether it is shorter to go clockwise or counterclockwise. With a doubling trick, one can first take 1 step clockwise from the starting point. If the destination is not reached, turn back and go counterclockwise from the starting point for 2 steps. If the destination is still not reached, turn back and go clockwise from the starting point for 4 steps and so on. Within $O(k)$ steps in total, one can find the destination if the destination is k steps away.

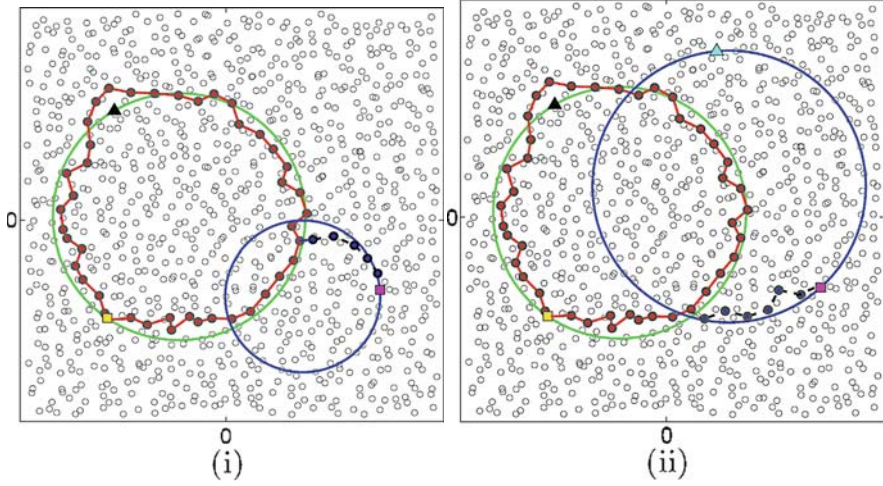


Fig. 5.19 Dark triangle denotes the hashed location; the red paths denote producer replication curves; dashed blue paths denote retrieval curves; yellow square denotes one producer and magenta square denotes one consumer. (i) Consumer latitude curve. (ii) Consumer great circle curve

3. *Aggregated data retrieval rule:* Since any closed curve that separates the hashed location h from its antipodal point $\bar{h}$ will intersect all the replication curves with the same data type, a consumer now has great flexibility to choose a retrieval curve for a set of data types $\{T_i\}$, $i = 1, \dots, m$. In particular, a consumer can follow a data retrieval curve that, for each data type T_i ,

- Either goes through the hashed location $h = h(T_i)$ or $\bar{h}$, where the aggregates are computed and stored
- Or is a closed curve that separates h from $\bar{h}$, collects all the relevant data and computes the aggregates

The above retrieval rule does not specify a unique retrieval curve but allow infinitely many possibilities. Thus multiple consumers searching for the same data type may choose, by their own decisions, different routes. This flexibility of data retrieval rule enables load balanced traffic patterns and routing robustness.

4. *Full power data retrieval rule:* The consumer travels along any great circle and is able to retrieve *all* the data stored in the network, because any two great circles intersect. An example is shown in Fig. 5.19.

The spherical double rulings scheme, with modestly increased replication, is more robust to node failures. With the flexibility in retrieval curves, the rendezvous node is no longer a bottleneck since retrieval curves may not necessarily visit it. In addition, the data storage scheme admits a local recovery scheme upon node failures. If the sensors in a certain region are destroyed, then all the relevant data are stored on the boundary and thus can be locally recovered.

5.5.2.3 Rumor Routing

Rumor routing, proposed by Braginsky and Estrin [11], can be considered as a probabilistic double rulings scheme. Information producer takes a walk and leaves data pointers on the trail. A consumer travels along another walk hoping to encounter one of the data pointers. The retrieval curve has a probability to intersect with the producer curve. Thus the consumer sends out enough retrieval walks to have a sufficiently high probability to meet with one of the event curves.

If the paths taken by the producer and consumers are along straight lines, the probability that two random lines intersect inside the sensor field has a constant probability. If the paths taken follow a random walk, the probability that two random walks intersect is lower. The algorithm is randomized with a small probability to fail to discover the producer curve. The attractive property is its robustness to node failures by the built-in randomization in the algorithm design.

5.5.2.4 Double Rulings with Virtual Coordinates

Double rulings schemes can also be defined in the virtual coordinate space. In fact, many virtual coordinates naturally admit two sets of rulings orthogonal to each other, e.g., the horizontal lines and vertical lines in a Cartesian coordinate space; the rays emitting from the origin and the cocentric circles in a polar coordinate space. Fang et al. [27] used double rulings scheme for data-centric routing in a general sensor field with nontrivial topology. The idea is to combine double rulings with GHTs on the two-level glider [26]. Recall that in glider, a set of landmarks are selected and the sensors are partitioned into Voronoi tiles, with each tile containing the nodes closest to the same landmark. The top level of the hierarchy is the combinatorial Delaunay graph that describes the adjacency information of the Voronoi tiles. Conceptually, the landmark-based information storage and retrieval scheme has two levels a distributed hash table for information storage at the tile adjacency graph level and a double-ruling scheme at the lower level (inside each tile), which ensures information retrieval within each tile.

In particular, on the top level a data type is hashed to a tile instead of a single node. On the basis of the tile adjacency graph, the shortest path tree rooted at the hashed tile can be computed at each node. All producers and consumers of the same content proceed to the hashed tile following this common shortest path tree. For a producer, data are replicated inside all the tiles (not all the nodes) along the way from where the producer resides to the hashed tile (which we call the *replication path*). The information consumer proceeds toward the hashed tile and checks each tile on its way for the desired data (which we call the *retrieval path*), returns when the retrieval path meets the replication path. At the bottom level, i.e., inside each tile, data storage and retrieval are implemented by a landmark-based double rulings scheme. In the worst case, the consumer visits the final hashed tile to retrieve the

desired data. Often on the way to the final tile, the consumer may hit a tile that already has the desired information and stops moving further.

5.6 Location Services and Hierarchical Routing Schemes

In geographical routing or routing with virtual coordinates, the source routes a message to a destination with the *name* (the geographical location or the virtual coordinates) of the destination. In certain applications such as routing to a geographical location or region, the source knows the destination's location already. For other applications, such as routing to a node with a particular ID, the source needs to obtain the location or the name of the destination, via a *location service*. A location service supports queries from any node for the name of any other node based on its ID. For the Internet, the mapping between host names and IP addresses is implemented by centralized servers called the DNS. In sensor networks, location service is preferred to be implemented in a distributed fashion, to avoid overloading the sensors identified as location servers.

Location service can be viewed as a special case of data-centric routing. In particular, the data of interest is the name of a node, the producer of this data is the node itself. Data-centric routing or data-centric storage is used to disseminate and store the names of nodes in a distributed way such that any other node can query for this information.

In this section, we cover a few hierarchical routing schemes. The routing schemes described in Sect. 5.5 are more or less “flat”, exploiting the two-dimensional embedding of sensor nodes in the physical space. Alternatively hierarchies can be built on the nodes for both point-to-point routing and data-centric storage and information retrieval. Typically when we think of a hierarchical data structure, we are often concerned about load balancing issue and the single point of failure at the root of the hierarchy. The following schemes exploit a few novel ideas to disseminate work load and avoid overloading high-level nodes on the hierarchy.

5.6.1 Landmark Hierarchy

5.6.1.1 Landmark Hierarchy

Probably as the first hierarchical routing scheme, Tsuchiya [100] proposed a landmark hierarchy for node naming and routing. Every node is a landmark and the nodes are organized into landmarks of different levels. A landmark at a level i has a radius r_i . Nodes within hop count distance r_i from a landmark at a level i have routing table entries to this landmark. The radii r_i s are increasing as the level i is increasing. At the highest level h , there are a small number of “global” landmarks such that every node has a routing table entry for them. In other words, every node

knows how to route to these global landmarks, and they know how to route to nearby “local” landmarks at different resolution.

To construct the landmark hierarchy, every node is a landmark at level 0. Some landmarks of level $i - 1$ are selected to be at level i such that every landmark at level $i - 1$ has at least one level- i landmark within distance r_{i-1} . A node may be a landmark at multiple levels. Each node p maintains routing table entries for all the landmarks ℓ such that p is within distance r_i of ℓ if ℓ is a level- i landmark.

The address of a node p is taken as a chain of landmarks $l_0, l_1, \dots, l_h$ such that the landmark ℓ_{i+1} is within distance r_i from ℓ_i , and $\ell_0 = p$. In other words, the landmark ℓ_i has a routing table entry to ℓ_{i-1} . Routing is based on the landmark address in a greedy fashion. To route to a node p , a node checks its routing table entry for p 's addressing landmarks and takes the lowest level landmark ℓ_i (in the worst case the global landmark ℓ_h). Then the packet is delivered to the next hop toward ℓ_i . This is repeated until the message is delivered to the destination. When the packet is far away from the destination, it is guided by the high-level addressing landmark of the destination and the routing information is refined as the packet gets close to the destination. A few remarks are in place.

- The addressing landmarks are not necessarily unique. A node may have multiple valid addresses according to the naming scheme.
- The routing scheme guarantees message delivery but the path may not be the shortest path from the source to destination.
- The paths to a destination p do not necessarily go through p 's addressing landmarks.

The landmark routing makes use of a landmark hierarchy but routing takes advantages of the cross-branch links in the hierarchy, thus having better robustness to topological changes, compared with routing on a tree. This idea is further developed later in two aspects: refine the construction of the landmark hierarchy to obtain bounded routing path stretch (the ratio of the path length vs. the shortest path length); and use the landmark hierarchy for location service and address lookup [18, 34].

5.6.1.2 Discrete Center Hierarchy

Funke et al. [34] refined the hierarchy by taking a discrete center hierarchy [39] defined as follows. Similar to the landmark hierarchy above, each node is a level-0 center. The level- i centers are selected as a subset of the level $i - 1$ centers that satisfies both the *covering* and *packing* properties:

- Every level $i - 1$ center is within distance 2^i from at least one level i center
- Two level i centers are at least distance 2^i away from each other

The highest level is at most $\log_2 n$, as the covering radius doubles at every level. Each level $i - 1$ center picks one of the level i center within distance 2^i and denotes

it as its *parent*. Thus the address of a node p is determined by its ancestors in the hierarchy, $l_0, l_1, \dots, l_h$, in which $l_0 = P$ and l_i is the parent of l_{i+1} .

With the discrete center hierarchy, cross-branch links are also included to help routing. In particular, a center v at level i has its *cluster* $C(v)$ as the set of descendant nodes. A node u has a cluster $C(v)$ at level i as a *neighboring cluster* if there is a node q in $C(v)$ within distance $\alpha \cdot 2^{i+1}$ for a constant $\alpha > 0$. Each node maintains a routing table entry for each neighboring cluster at each level. Again a node has routing information to nearby neighboring clusters at different resolutions. The following properties can be said to the neighboring clusters.

- The distance from a node to its level i ancestor is bounded by 2^{i+1} . In particular, we follow the parent link and sum up its maximum possible length, we get $1 + 2 + 2^2 + \dots + 2^i \leq 2^{i+1}$.
- If we take $\alpha > 3$, a node in cluster $C(v)$ at level i has $C(w)$ as a neighboring cluster, where w is a child of v , w has level $i - 1$.

For a node u to route a packet to a destination node v , u checks its routing table for the lowest level cluster $C(w)$ that contains the destination. Then u routes the packet toward the cluster $C(w)$. When the packet arrives at the cluster $C(w)$, the same procedure is repeated at the next node. The new hierarchy guarantees that the route taken is within four times the shortest path length d_{uv} from source u to destination v . Indeed, say the cluster $C(w)$ at lowest level found in u 's routing table entry is at level k , then d_{uv} must be greater than $\alpha \cdot 2^k$ (otherwise u has a neighboring cluster at level $k - 1$ that contains the destination). The path taken from u to $C(w)$ is at most $\alpha \cdot 2^{k+1} \leq 2d_{uv}$. Once the packet arrives at $C(w)$ it is able to find a neighboring cluster at level strictly lower than k . Thus the path taken to arrive at the next lower level cluster is at most half the length of the first segment on the path, i.e., the path taken so far from u to $C(w)$. Thus the total path length is at most double the length of the first segment and is less than $4d_{uv}$.

In a sensor network setting, when the sensors are reasonably uniformly distributed, the communication graph typically has constant doubling dimension [47], i.e., a ball of radius R can be covered by a constant number of balls of radius $R/2$. In this case, the hierarchy has depth $O(\log n)$ and each node has a constant number of neighboring clusters. Therefore, the number of entries in the routing table for each node is $O(\log n)$ in the worst case and on average a constant number.

The landmark hierarchy can also support data-centric routing and distributed location services. In particular, a data item is hashed and stored at certain other nodes called the *data servers*. Specifically, a producer u has its ancestors at each level, denoted as $l_0 = u, l_1, \dots, l_h$. For each level i , the producer sends the data to each of u 's neighboring clusters at that level. For one of such cluster $C(p)$, the data are hashed by its data type to one node q inside this cluster. Again the number of copies a data item is stored in the network is at most $O(\log n)$.

Storing data in the landmark hierarchy enables the consumer to find the data item in a distance sensitive fashion, i.e., a producer with distance d from the consumer is able to retrieve the data within cost $4d$. In particular, when a consumer v wants to find a data item, it starts to visit the data server in the clusters of its ancestors at

level i , with i increasing from 0, until the data item is found or the highest level is reached. Intuitively, the data storage scheme places more data servers in nearby clusters and fewer servers in far away but large clusters. The retrieval scheme starts to search in local neighborhood until it discovers a nearby data server.

At last we remark that similar idea of using a landmark hierarchy for routing and location service was also explored in [4, 17, 18]. These constructions use different ways to construct the hierarchy and admit different quality bounds on the routing performance.

5.6.2 Routing and Location Services in Mobile Networks

In a mobile network, the landmark hierarchies described in the previous subsection are not appropriate as the structure is dependent on the network distance which changes frequently as the nodes move around. Geographical routing, on the other hand, is robust in a mobile network given that the source knows the location of the destination. As the nodes constantly move around, we will need a location service for the mobile nodes such that any node can query for the current location of any other node. Notice that in implementing this location service we will need some routing protocol to deliver the query messages. Thus routing and location service appear to be a chicken and egg problem that will need to be solved all at once in a collaborative framework. The grid location service (GLS), proposed by Li et al. [65], represents an example of such a protocol.

GLS implements a distributed location service in which a node serves as a *location server* and stores location information for other nodes. A node's location is stored in multiple other location servers. As a node moves around and changes its location, it will update its location servers of its new location. Any other node looking for the location of a node with only the ID of the node is accomplished with pretty much the same protocol as the node uses to select its location server in the first place. GLS uses consistent hashing [52], originally proposed for hierarchical hashing of web pages in the Internet, for location server assignment.

We assume the nodes know their geographical locations via GPS devices for example. The domain in which the nodes move around is partitioned recursively as a quadtree of depth k such that the smallest square contains only one node. The smallest square is called an order-1 square. The bounding box is called an order- k square and partitioned into 4 equal-size order $k - 1$ squares. In fact, other type of balanced partitioning can also be used here. The selection of location servers follows the same philosophy as the previous section. A node u stays in a unique square of order i , for each i from 1 to k . u selects a location server in each of the three other sibling order- i squares of the one containing u . In particular, inside a square S , the location server is chosen as the node *closest* to the ID of u , defined as the one with *least ID greater than u* . The ID space is circular. For example, ID 2 is closer to 17 than 7 is.

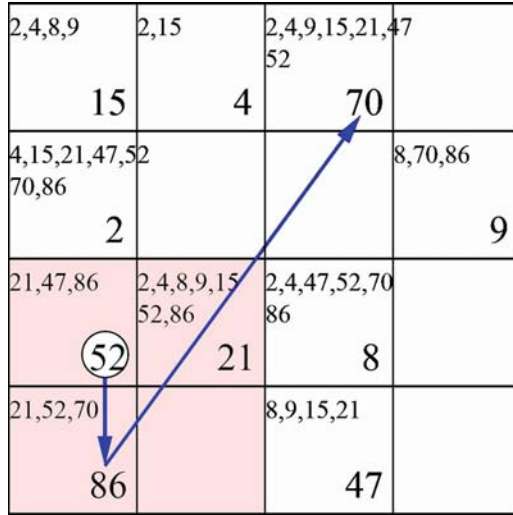


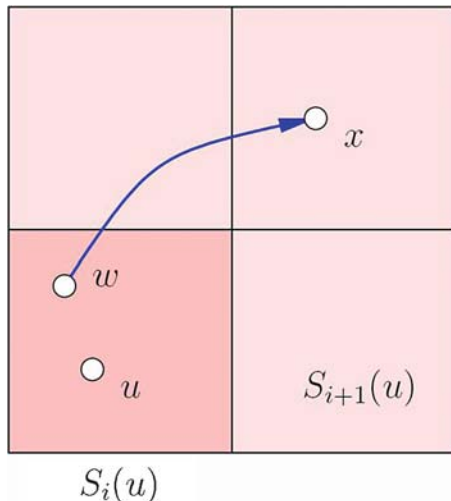
Fig. 5.20 An example of GLS. In each small square the number at lower right is the ID of the node inside this square. The numbers on the *upper left* are the IDs of other nodes for which this node is the location server. The *arrows* show that a query at 52 looking for the location of a node with ID 70

To perform a query for the location of v , u knows only the ID of v and does not know the location servers of v . In fact v does not know its location servers either. Now recall that u is also the location server of some other nodes. u will simply send a request by geographical routing to the node closest to v for which u has location information. Say this node is w . This request can be delivered by geographical routing as u has the location information of w . Now w does the same thing until the message reaches a location server of v , from where the message can then be delivered to v via geographical routing. An example is shown in Fig. 5.20.

The correctness of the scheme follows the following claim that the query visits the node closest to v in u 's order- i square $S_i(u)$, with increasing i . Thus if v is inside node u 's order k square, then the closest node to v is v itself. Thus the message will be delivered to v . To see that the claim is true, we show by induction. At level $i = 1$, it is trivially true as u is the only node in its order-1 square. Now assume that the message arrives at a node w which is the closest node to v in u 's order i square $S_i(u)$. This says that in $S_i(u)$, there is no other nodes in between the ID of v and w . Now suppose that the node closest to v in $S_{i+1}(u)$ is x . If x is w , then the claim is true trivially. If x is not w , x stays in a sibling square of $S_i(u)$. x will choose its location server in $S_i(u)$ as the node closest to x , which must be w . Thus w is the location server of the node closest to v in $S_{i+1}(u)$. The routing protocol will forward the message to x . See Fig. 5.21 for an illustration.

When a node moves around, it will update its location at its location servers. This update operation is essentially the same as the query algorithm. When a node

Fig. 5.21 The node closest to v in $S_{i+1}(u)$, denoted by x , if not equal to w , must choose w as its location server in $S_i(u)$. Thus w has x 's location and the message is delivered to x



u wants to update the location server at a square S , it sends a message to that square with geographical routing. Once the message arrives at S , the first node will start with a query for the node u inside S . As shown by the query algorithm, this message is eventually forwarded to the node closest to u inside S before it leaves this square, which is precisely the location server for u . Thus location information is updated at this location server.

To initialize the system, the location servers are updated from bottom up. The nodes first locate their location servers in order 1 sibling squares. After this is done, the nodes use the location update scheme to locate the location servers in order-2 sibling squares, and so on. Thus no flooding is needed to set up the system.

The combination of location service with geographical routing, with the help of consistent hashing is a very neat idea that requires no global coordination. The scheme is later improved to have worst-case bounded performance in terms of location update and distance-sensitive query (such as LLS by Abraham et al. [1] and MLS by Flury and Wattenhofer [31]). The idea used in LLS is to have a node publish its location along a spiral that exponentially increases in distance. Similarly, the lookup operation also takes a spiral from the query's location. When the lookup spiral visits a square in which the location update has visited, the query will return the location information. The intuition is that the query spiral will intersect the publishing spiral within a traveling distance $O(d)$ if the node is of distance d away.

5.7 Thoughts for Practitioners

For practitioners, the selection of routing algorithms may depend on the following considerations.

- Sensor node capacity
 - Power supply: battery power or long-lasting power supplies
 - Size of on-board memory
 - Radio characteristics (radio range, directional antennas or omni-directional antennas, data lost rate)
- Network scale and deployment setting
 - Size of the network and the size of the deployment region
 - Node density
 - Node distribution uniformity (existence of routing voids or not)
 - Node mobility
 - Homogeneous nodes or heterogeneous nodes
 - Accessibility of the network
 - Availability of sensor node locations
- Application requirements
 - Traffic pattern (many-to-one data collection, one-to-many data dissemination, point-to-point routing, and data-centric routing)
 - Traffic load (frequency of message delivery)
 - Expected network operational time
 - Data packet size and format
 - Quality-of-service requirement on latency and throughput

5.8 Directions for Future Research

The work surveyed in this chapter revealed the power of using geometric abstractions in sensor network routing. There are still directions for future research on many fundamental problems.

- Communication models: modeling wireless communication channels has a fundamental influence in routing algorithm design. Simplified models such as unit disk graphs have been adopted widely in the algorithm and theory community as it is simple yet embraces a rich set of geometric structures. However, the deviation from realistic scenarios makes some algorithms with unit disk graph model fail in practice. More realistic models have been proposed and adopted. But it still remains a challenge to develop different communication models that are more elaborative and realistic, yet still reveals key insights and admit efficient routing algorithm design.
- Supporting infrastructure support: applying the geometric ideas in routing algorithm design often requires some supporting architecture components such as localization, synchronization, and topology understanding.

- **Mobility:** most algorithms covered in this chapter assume static sensor networks. With node mobility and/or heterogeneity, new ideas are needed on what geometric structure can we abstract from the node mobility and how does that help with routing information in the network efficiently.
- **Geometry of the sensor data:** when a sensor network monitors a physical signal landscape, there is natural correlation between data from sensor nodes, both in the spatial and temporal spectrum. Abstracting and utilizing the geometry of sensor data can help with data compression, aggregation, and validation in routing algorithms.
- **Theoretical foundation:** a number of the geometric algorithms discussed in this chapter use ideas from the continuous geometric domain and apply an analog of the ideas in the discrete network. Investigation of rigorous definition of the geometric concepts in the discrete network remains an open problem.

5.9 Conclusions

In the past few years, geometric ideas have been used extensively in the design of wireless sensor networks. This chapter covers routing algorithms that make use of various aspects of “sensor network geometry”: the nodes’ geographical locations, such as geographical routing; as well as the global shape and topology of a sensor field.

Terminologies

Network localization. In a wireless network, network localization finds the geographical location of all the nodes. A network location algorithm can be anchor-based or anchor-free. Anchor-based algorithms assume known location information for some of the nodes (called anchors) in the network. An anchor-free algorithm does not assume the knowledge of any location information and returns the relative positions of the network.

Geographical routing. In a wireless ad hoc network, geographical routing is a routing algorithm that uses the nodes’ geographical locations to choose a neighbor as the next hop.

Virtual coordinates. Each node in a wireless sensor network is assigned a virtual coordinate as its identifier in the routing algorithm. A virtual coordinate routing algorithm makes routing decisions with the virtual coordinates of the source, its neighbors, and the destination. The size of the virtual coordinate is typically a constant or a small polylogarithmic function of the network size.

Greedy embedding. Given a graph, embed it in a d -dimensional Euclidean space such that for each pair of node p, q , there is a neighbor of p whose distance to

the destination is smaller than the Euclidean distance between p and q . Such an embedding is called a greedy embedding.

Landmark routing. A subset of nodes in a wireless sensor network are selected as landmarks such that the hop count distances to these landmarks are used to identify other nodes as well as route messages between them.

Data centric routing. The destination of the message is not specified as a fixed destination node, but rather defined as the sensor node(s) that has the information as requested in the message. A data-centric routing algorithm needs to discover the nodes holding data of interest as well as route the message to these nodes. The nodes with data are often called sources or information producers. The nodes who pose queries for the data are called sinks or information consumers.

Geographical hash tables. It is a data-centric routing scheme. A source node will take a hash function on the data type and deliver the data to a geographical hash location. The nodes close to the hashed location serve as rendezvous and store the data. A sink node searching for this type of data will route the query to the same geographical location and retrieve the data from the rendezvous nodes.

Double rulings. It is a data-centric routing scheme. The source node will hash data along a data storage curve (possibly dependent on the source location and the data type) in the network. The sink node will hash data along a data retrieval curve (possibly dependent on the sink location and the data type) in the network. If each data storage curve intersects with each data retrieval curve, the data retrieval mechanism guarantees successful data discovery.

Location service. For geographical routing, a source node needs to know the destination's location to send a message to it. Location service provides a map between each node and its geographical location. Location service can be implemented at a central location server or at distributed location servers.

Landmark hierarchy. All nodes are landmarks at different levels of the landmark hierarchy. A node is given a name according to the network distances to some other landmarks in the landmark hierarchy. Routing decisions are made with the name of the source and the destination. The landmark hierarchy can be considered as multilevel landmark routing.

Questions

1. In network localization, suppose that k nodes out of a total number of n nodes are anchor nodes. There are m edges in the graph. How large does m have to be such that there is no degree of freedom in the system?
2. Prove the following two claims that will show both relative neighborhood graph and Gabriel graph on a set of points P in the plane are planar, connected subgraphs of unit disk graphs.
 - (a) Prove that the relative neighborhood graph is a subgraph of the Gabriel Graph.

- (b) Prove that the Gabriel graph is a subgraph of the Delaunay triangulation on the same set of nodes.

For a set of point P in the plane, the Voronoi diagram partitions the plane into connected cells such that all the points in the same cell has the same closest point in P . The Delaunay triangulation is the dual graph of the Voronoi diagram such that there is an edge uv , if the cells of u, v are adjacent. The Delaunay triangulation has what is called “empty-circle property”: for any Delaunay edge, one can find a circle that goes through the two endpoints of the edge and has no other points inside. The converse is also true: if all edges in a graph has this empty circle property, the graph is a Delaunay triangulation.

The Delaunay triangulation is known to be a planar graph. Thus there are no crossing edges in either relative neighborhood graph or Gabriel graph.

- (c) Prove that the relative neighborhood graph contains the minimum spanning tree. Thus both the relative neighborhood graph and the Gabriel graph are connected.
3. Prove that a Delaunay triangulation of a set of point in the plane admits a greedy embedding.
 4. Show that a star with 7 leaf nodes does not have a greedy embedding in the plane.
 5. Prove that for any graph with a subset of vertices pinned down, the rubberband representation (defined in Sect 5.4.1.1) is uniquely determined. (Hint: show that the rubberband embedding minimizes a convex energy function.)
 6. Prove for the case of Euclidean plane that greedy routing algorithm used in intratile routing of GLIDER by minimizing the centered virtual distance to the destination is free of local minimum, when the landmarks are not collinear.
 7. Elaborate how to incorporate node mobility in geographical routing.
 8. Prove that in landmark-based routing, if we choose at least three landmarks that are not collinear and define the virtual coordinate of any other point in the Euclidean plane as the Euclidean distances to these landmarks, the virtual coordinate for each point is unique. Also give an example that this may not be true in a discrete wireless network (modeled by unit disk graph).
 9. Compare the pros and cons of geographical hash tables and double rulings scheme for data-centric routing, in terms of storage requirement, query latency, robustness to node failures, etc.
 10. Elaborate how to handle node insertion in the Grid location service (GLS).

References

1. I. Abraham, D. Dolev, and D. Malkhi. LLS: A locality aware location service for mobile ad hoc networks. In *DIALM-POMC '04: Proceedings of the 2004 joint workshop on Foundations of mobile computing*, 2004. ACM Press, New York, pages 75–84.

2. P. Angelini, F. Frati, and L. Grilli. An algorithm to construct greedy drawings of triangulations. In *Proceedings of the 16th International Symposium on Graph Drawing*, pages 26–37, 2008.
3. J. Aspnes, D. Goldenberg, and Y. R. Yang. On the computational complexity of sensor network localization. In *The First International Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS)*, pages 32–44, 2004.
4. B. Awerbuch and D. Peleg. Concurrent online tracking of mobile users. In *SIGCOMM '91: Proceedings of the conference on Communications architecture & protocols*, 1991. ACM Press, New York, pages 221–233.
5. M. Badoiu, E. D. Demaine, M. T. Hajiaghayi, and P. Indyk. Low-dimensional embedding with extra information. In *SCG '04: Proceedings of the twentieth annual symposium on Computational geometry*, 2004. ACM Press, New York, pages 320–329.
6. P. Bahl and V. N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *IEEE INFOCOM*, Volume 2, pages 775–784, 2000.
7. P. Biswas and Y. Ye. Semidefinite programming for ad hoc wireless sensor network localization. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, pages 46–54, 2004.
8. L. Blazević, L. Buttyán, S. Capkun, S. Giordano, H. Hubaux, and J. L. Boudec. Self-organization in mobile ad hoc networks: The approach of terminodes. *IEEE Communications Magazine*, pages 166–175, 2001.
9. I. Borg and P. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer-Verlag, Berlin 1997.
10. 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.
11. D. Braginsky and D. Estrin. Rumor routing algorithm for sensor networks. In *Proc. of the 1st ACM Int'l Workshop on Wireless Sensor Networks and Applications (WSNA)*, pages 22–31, September 2002.
12. H. Breu and D. G. Kirkpatrick. Unit disk graph recognition is NP-hard. *Computational Geometry: Theory and Applications*, 9(1–2):3–24, 1998.
13. J. Bruck, J. Gao, and A. Jiang. Localization and routing in sensor networks by local angle information. In *Proceedings of the Sixth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'05)*, pages 181–192, May 2005.
14. J. Bruck, J. Gao, and A. Jiang. MAP: Medial axis based geometric routing in sensor networks. *Wireless Networks*, 13(6):835–853, 2007.
15. M. Caesar, M. Castro, E. B. Nightingale, G. O'Shea, and A. Rowstron. Virtual ring routing: Network routing inspired by dhds. In *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, 2006. ACM Press, New York, pages 351–362.
16. A. Caruso, A. Urpi, S. Chessa, and S. De. GPS free coordinate assignment and routing in wireless sensor networks. In *Proceedings of the 24th Conference of the IEEE Communication Society (INFOCOM)*, Volume 1, pages 150–160, March 2005.
17. H. T.-H. Chan, A. Gupta, B. M. Maggs, and S. Zhou. On hierarchical routing in doubling metrics. In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, 2005. Society for Industrial and Applied Mathematics, Pennsylvania, pages 762–771.
18. B. Chen and R. Morris. L+: Scalable landmark routing and address lookup for multi-hop wireless networks. Technical Report MIT-LCS-TR-837, Massachusetts Institute of Technology, 2002.
19. M. B. Chen, C. Gotsman, and C. Wormser. Distributed computation of virtual coordinates. In *SCG '07: Proceedings of the twenty-third annual symposium on Computational geometry*, 2007. ACM Press, New York, pages 210–219.
20. Y.-C. Cheng, Y. Chawathe, A. LaMarca, and J. Krumm. Accuracy characterization for metropolitan-scale wi-fi localization. In *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*, 2005. ACM Press, New York, pages 233–245.

21. H. S. M. Coxeter. *Introduction to Geometry*. Wiley, New York, 2nd edition, 1969.
22. R. Dhandapani. Greedy drawings of triangulations. In *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, 2008.
23. L. Doherty, L. E. Ghaoui, and S. J. Pister. Convex position estimation in wireless sensor networks. In *IEEE Infocom*, Volume 3, pages 1655–1663, April 2001.
24. A. Efrat, C. Erten, D. Forrester, A. Iyer, and S. G. Kobourov. Force-directed approaches to sensor localization. In *Proceedings of the 8th Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 108–118, 2006.
25. Q. Fang, J. Gao, and L. Guibas. Locating and bypassing routing holes in sensor networks. In *Mobile Networks and Applications*, Volume 11, pages 187–200, 2006.
26. Q. Fang, J. Gao, L. Guibas, V. de Silva, and L. Zhang. GLIDER: Gradient landmark-based distributed routing for sensor networks. In *Proceedings of the 24th Conference of the IEEE Communication Society (INFOCOM)* Volume 1, pages 339–350, March 2005.
27. Q. Fang, J. Gao, and L. J. Guibas. Landmark-based information storage and retrieval in sensor networks. In *The 25th Conference of the IEEE Communication Society (INFOCOM'06)*, pages 1–12, April 2006.
28. S. P. Fekete, M. Kaufmann, A. Kröller, and N. Lehmann. A new approach for boundary recognition in geometric sensor networks. In *Proceedings 17th Canadian Conference on Computational Geometry*, pages 82–85, 2005.
29. S. P. Fekete, A. Kröller, D. Pfisterer, S. Fischer, and C. Buschmann. Neighborhood-based topology recognition in sensor networks. In *ALGOSENSORS*, Volume 3121 of *Lecture Notes in Computer Science*, 2004 Springer, Berlin, pages 123–136.
30. G. G. Finn. Routing and addressing problems in large metropolitan-scale internetworks. Technical Report ISU/RR-87-180, ISI, March 1987.
31. R. Flury and R. Wattenhofer. MLS: An efficient location service for mobile ad hoc networks. In *MobiHoc '06: Proceedings of the seventh ACM international symposium on Mobile ad hoc networking and computing*, 2006. ACM press, New York, pages 226–237.
32. R. Fonesca, S. Ratnasamy, J. Zhao, C. T. Ee, D. Culler, S. Shenker, and I. Stoica. Beacon vector routing: Scalable point-to-point routing in wireless sensor networks. In *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI)*, pages 329–342, May 2005.
33. S. Funke. Topological hole detection in wireless sensor networks and its applications. In *DIALM-POMC '05: Proceedings of the 2005 Joint Workshop on Foundations of Mobile Computing*, 2005. ACM Press, New York, pages 44–53.
34. S. Funke, L. J. Guibas, A. Nguyen, and Y. Wang. Distance-sensitive routing and information brokerage in sensor networks. In *IEEE International Conference on Distributed Computing in Sensor System (DCOSS'06)*, pages 234–251, June 2006.
35. S. Funke and C. Klein. Hole detection or: “How much geometry hides in connectivity?”. In *SCG '06: Proceedings of the twenty-second annual symposium on Computational geometry*, pages 377–385, 2006.
36. S. Funke and N. Milosavljević. Guaranteed-delivery geographic routing under uncertain node locations. In *Proceedings of the 26th Conference of the IEEE Communications Society (INFOCOM'07)*, pages 1244–1252, May 2007.
37. S. Funke and N. Milosavljević. Network sketching or: “how much geometry hides in connectivity? - part II”. In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 2007. Society for Industrial and Applied Mathematics, Pennsylvania, pages 958–967.
38. D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. Complex behavior at scale: An experimental study of low-power wireless sensor networks. Technical Report UCLA/CSD-TR 02-0013, UCLA, 2002.
39. J. Gao, L. Guibas, and A. Nguyen. Deformable spanners and their applications. *Computational Geometry: Theory and Applications*, 35(1–2):2–19, 2006.
40. J. Gao, L. J. Guibas, S. Y. Oudot, and Y. Wang. Geodesic Delaunay triangulations and witness complexes in the plane. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA'08)*, 571–580, January 2008.

41. R. Ghrist and A. Muhammad. Coverage and hole-detection in sensor networks via homology. In *Proceedings the 4th International Symposium on Information Processing in Sensor Networks (IPSN'05)*, pages 254–260, 2005.
42. D. K. Goldenberg, P. Bihler, Y. R. Yang, M. Cao, J. Fang, A. S. Morse, and B. D. O. Anderson. Localization in sparse networks using sweeps. In *MobiCom '06: Proceedings of the 12th annual international conference on Mobile computing and networking*, 2006, ACM Press, New York, pages 110–121.
43. M. T. Goodrich and D. Strash. Succinct greedy geometric routing in r_2 . Technical report on arXiv:0812.3893, 2008.
44. C. Gotsman and Y. Koren. Distributed graph layout for sensor networks. In *Proceedings of the International Symposium on Graph Drawing*, pages 273–284, September 2004.
45. Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker. Lazy cross-link removal for geographic routing. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, 2006. ACM Press, New York, pages 112–124.
46. B. Greenstein, D. Estrin, R. Govindan, S. Ratnasamy, and S. Shenker. DIFS: A distributed index for features in sensor networks. In *Proceedings of First IEEE International Workshop on Sensor Network Protocols and Applications*, pages 163–173, Anchorage, Alaska, May 2003.
47. A. Gupta, R. Krauthgamer, and J. R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *FOCS '03: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, 2003. IEEE Computer Society, Washington, page 534–543.
48. A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster. The anatomy of a context-aware application. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, 1999, ACM Press, New York, pages 59–68.
49. B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. *Global Positioning Systems: Theory and Practice*. 5th edition, Springer, Berlin, 2001.
50. C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *ACM Conference on Mobile Computing and Networking (MobiCom)*, pages 56–67, 2000.
51. R. Jain, A. Puri, and R. Sengupta. Geographical routing using partial information for wireless ad hoc networks. *IEEE Personal Communications*, 8(1):48–57, Feb. 2001.
52. D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, 1997. ACM Press, New york, pages 654–663.
53. B. Karp and H. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 243–254, 2000.
54. Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker. Geographic routing made practical. In *Proceedings of the Second USENIX/ACM Symposium on Networked System Design and Implementation (NSDI 2005)*, May 2005.
55. Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker. On the pitfalls of geographic face routing. In *DIALM-POMC '05: Proceedings of the 2005 joint workshop on Foundations of mobile computing*, 2005. ACM Press, New York, pages 34–43.
56. R. Kleinberg. Geographic routing using hyperbolic space. In *Proceedings of the 26th Conference of the IEEE Communications Society (INFOCOM'07)*, pages 1902–1909, 2007.
57. E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. In *Proceedings 11th Canadian Conference on Computational Geometry*, pages 51–54, 1999.
58. P. Krishnan, A. S. Krishnakumar, W.-H. Ju, C. Mallows, and S. Ganu. A system for LEASE: System for location estimation assisted by stationary emitters for indoor RF wireless networks. In *IEEE Infocom*, Volume 2, pages 1001–1011, Hongkong, March 2004.
59. A. Krölller, S. P. Fekete, D. Pfisterer, and S. Fischer. Deterministic boundary recognition and topology extraction for large sensor networks. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1000–1009, 2006.

60. F. Kuhn, T. Moscibroda, and R. Wattenhofer. Unit disk graph approximation. In *Proceedings of the 2004 Joint Workshop on Foundations of Mobile Computing*, pages 17–23, 2004.
61. F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. Geometric ad-hoc routing: Of theory and practice. In *Proceedings 22nd ACM International Symposium on the Principles of Distributed Computing (PODC)*, pages 63–72, 2003.
62. F. Kuhn, R. Wattenhofer, and A. Zollinger. Asymptotically optimal geometric mobile ad-hoc routing. In *Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 24–33, 2002.
63. A. M. Ladd, K. E. Bekris, A. Rudys, L. E. Kavraki, D. S. Wallach, and G. Marceau. Robotics-based location sensing using wireless ethernet. In *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, 2002. ACM Press, New York, pages 227–238.
64. T. Leighton and A. Moitra. Some results on greedy embeddings in metric spaces. In *Proceeding of the 49th IEEE Annual Symposium on Foundations of Computer Science*, pages 337–346, October 2008.
65. J. Li, J. Jannotti, D. Decouto, D. Karger, and R. Morris. A scalable location service for geographic ad-hoc routing. In *Proceedings of 6th ACM/IEEE International Conference on Mobile Computing and Networking*, pages 120–130, 2000.
66. Z. Li, W. Trappe, Y. Zhang, and B. Nath. Robust statistical methods for securing wireless localization in sensor networks. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, Piscataway, NJ, USA, 2005. IEEE Press, New York, pages 91–98.
67. N. Linial, L. Lovász, and A. Wigderson. Rubber bands, convex embeddings and graph connectivity. *Combinatorica*, 8(1):91–102, 1988.
68. X. Liu, Q. Huang, and Y. Zhang. Combs, needles, haystacks: Balancing push and pull for discovery in large-scale sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, 2004. ACM Press, New York pages 122–133.
69. S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: A tiny aggregation service for ad-hoc sensor networks. *SIGOPS Operating Systems Review*, 36(SI):131–146, 2002.
70. Y. Mao, F. Wang, L. Qiu, S. S. Lam, and J. M. Smith. S4: Small state and small stretch routing protocol for large wireless sensor networks. In *Proceedings of the 4th USENIX Symposium on Networked System Design and Implementation (NSDI 2007)*, April 2007.
71. D. Moore, J. Leonard, D. Rus, and S. Teller. Robust distributed network localization with noisy range measurements. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, 2004. ACM Press, New York, pages 50–61.
72. A. Nasipuri and R. E. Najjar. Experimental Evaluation of an Angle Based Indoor Localization System. In *Proceedings of the 4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, pages 1–9, Boston, MA, April 2006.
73. J. Newsome and D. Song. GEM: Graph embedding for routing and data-centric storage in sensor networks without geographic information. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003. ACM Press, New York, pages 76–88.
74. A. Nguyen, N. Milosavljevic, Q. Fang, J. Gao, and L. J. Guibas. Landmark selection and greedy landmark-descent routing for sensor networks. In *Proceedings of the 26th Conference of the IEEE Communications Society (INFOCOM'07)*, pages 661–669, May 2007.
75. D. Niculescu and B. Nath. Ad hoc positioning system (APS). In *IEEE GLOBECOM*, pages 2926–2931, 2001.
76. D. Niculescu and B. Nath. Ad hoc positioning system (APS) using AOA. In *IEEE INFOCOM*, Volume 22, pages 1734–1743, March 2003.
77. D. Niculescu and B. Nath. Error characteristics of ad hoc positioning systems (APS). In *MobiHoc '04: Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 20–30, 2004.

78. D. Niculescu and B. Nath. VOR base stations for indoor 802.11 positioning. In *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*, 2004. ACM Press, New York, pages 58–69.
79. C. H. Papadimitriou and D. Ratajczak. On a conjecture related to geometric routing. *Theoretical Computer Science*, 344(1):3–14, 2005.
80. N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *MobiCom '00: Proceedings of the 6th ACM Annual International Conference on Mobile Computing and Networking*, pages 32–43, 2000.
81. A. Rao, C. Papadimitriou, S. Shenker, and I. Stoica. Geographic routing without location information. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, 2003. ACM Press, New York, pages 96–108.
82. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, 2001. ACM Press, New York, pages 161–172.
83. S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. GHT: A geographic hash table for data-centric storage in sensornets. In *Proceedings 1st ACM Workshop on Wireless Sensor Networks and Applications*, pages 78–87, 2002.
84. A. I. T. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware '01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, London, UK, 2001. Springer-Verlag, Berlin, pages 329–350.
85. R. Sarkar, X. Zhu, and J. Gao. Double rulings for information brokerage in sensor networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 286–297, September 2006.
86. A. Savvides, C.-C. Han, and M. B. Strivastava. Dynamic fine-grained localization in *ad-hoc* networks of sensors. In *Proceedings 7th Annual International Conference on Mobile Computing and Networking (MobiCom 2001)*, Rome, Italy, July 2001. ACM Press, New York, pages 166–179.
87. A. Savvides, H. Park, and M. B. Strivastava. The n -hop multilateration primitive for node localization problems. *Mobile Networks and Applications*, 8(4):443–451, 2003.
88. K. Seada, A. Helmy, and R. Govindan. On the effect of localization errors on geographic face routing in sensor networks. In *IPSN '04: Proceedings of the third international symposium on Information processing in sensor networks*, 2004. ACM Press, New York, pages 71–80.
89. Y. Shang, W. Ruml, Y. Zhang, and M. P. J. Fromherz. Localization from mere connectivity. In *MobiHoc '03: Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 201–212, 2003.
90. S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin. Data-centric storage in sensornets. *SIGCOMM Computer Communication Review*, 33(1):137–142, 2003.
91. A. M.-C. So and Y. Ye. Theory of semidefinite programming for sensor network localization. In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, 2005. Society for Industrial and Applied Mathematics, Pennsylvania, pages 405–414.
92. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, 2001. ACM Press, New York, pages 149–160.
93. I. Stojmenovic. A routing strategy and quorum based location update scheme for ad hoc wireless networks. Technical Report TR-99-09, SITE, University of Ottawa, September, 1999.
94. I. Stojmenovic and X. Lin. Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 12(10):1023–1032, 2001.
95. I. Stojmenovic and X. Lin. Power-aware localized routing in wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 12(11):1122–1133, 2001.
96. H. Takagi and L. Kleinrock. Optimal transmission ranges for randomly distributed packet radio terminals. *IEEE Transactions on Communications*, 32(3):246–257, 1984.

97. R. Thomas and X. Yu. 4-connected projective-planar graphs are hamiltonian. *Journal of Combinatorial Theory Series B*, 62(1):114–132, 1994.
98. M. Thorup and U. Zwick. Approximate distance oracles. In *Proceedings ACM Symposium on Theory of Computing*, pages 183–192, 2001.
99. M. Thorup and U. Zwick. Compact routing schemes. In *SPAA '01: Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures*, 2001. ACM Press, New York, pages 1–10.
100. P. F. Tsuchiya. The landmark hierarchy: A new hierarchy for routing in very large networks. In *SIGCOMM '88: Symposium on Communications architectures and protocols*, 1988. ACM Press, New York, pages 35–42.
101. Y. Wang, J. Gao, and J. S. B. Mitchell. Boundary recognition in sensor networks by topological methods. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 122–133, September 2006.
102. R. Want, A. Hopper, V. Falcao, and J. Gibbons. The active badge location system. *ACM Transactions on Information Systems*, 10:91–102, January 1992.
103. M. Wattenhofer, R. Wattenhofer, and P. Widmayer. Geometric Routing without Geometry. In *12th Colloquium on Structural Information and Communication Complexity (SIROCCO)*, Le Mont Saint-Michel, France, May 2005.
104. K. Whitehouse and D. Culler. A robustness analysis of multi-hop ranging-based localization approximations. In *IPSN '06: Proceedings of the fifth international conference on Information processing in sensor networks*, 2006. ACM Press, New York, pages 317–325.
105. K. Whitehouse, C. Karlof, A. Woo, F. Jiang, and D. Culler. The effects of ranging noise on multihop localization: an empirical study. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, Piscataway, NJ, USA, 2005. IEEE Press, New York, pages 73–80.
106. F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang. A two-tier data dissemination model for large-scale wireless sensor networks. In *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, 2002. ACM Press, New York, pages 148–159.
107. M. Youssef, A. Agrawala, and U. Shankar. WLAN location determination via clustering and probability distributions. Technical report, University of Maryland, College Park, MD, March 2003.
108. F. Zhang, H. Li, A. A. Jiang, J. Chen, and P. Luo. Face tracing based geographic routing in nonplanar wireless networks. In *Proceedings of the 26th Conference of the IEEE Communications Society (INFOCOM'07)*, pages 2243–2251, May 2007.
109. B. Y. Zhao, J. D. Kubiawicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and. Technical report, Berkeley, CA, USA, 2001.
110. G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic. Impact of radio irregularity on wireless sensor networks. In *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, 2004. ACM Press, New York, pages 125–138.
111. G. Ziegler. *Lectures on Polytopes*. Springer-Verlag, Berlin 1995.