Chapter 15
Position-Based Routing in Wireless Ad Hoc and Sensor Networks

Nathalie Mitton, Tahiry Razafindralambo, and David Simplot-Ryl

Abstract Geometric routing protocols are a memoryless and scalable approach which uses position information for routing. Principles of geometric routing approaches are very simple. Every node is assumed to be aware of the location of itself, of its neighbors, and of the destination. Based only on these information, every node is able to perform a routing decision. The location can be determined by either geographic coordinates (we thus talk of geographic routing) or logical coordinates extracted from the environment. In the former case, location coordinates may be derived thanks to GPS or estimated thanks to any other positioning mean such as triangulation. In the latter case, a new coordinate system has to be built. This chapter reviews the main routing algorithms in every coordinate-based system, highlighting the strengths and weaknesses of each of them.

15.1 Introduction

Routing in wireless sensor networks is a challenging task. Many different approaches have been proposed in the literature. We can identify three main classes of routing protocols: (i) proactive routing such as OLSR [10], (ii) reactive routing such as AODV [29], and (iii) geometric routing or georouting. This latter approach is receiving more and more attention since it is a memoryless and scalable approach, unlike the two other ones. In addition, it better suits the constraints of wireless sensor networks in which memory and processing capacities are very low and in which the number of entities is expected to be very high. The idea of using position information for routing was first proposed in the 1980s in the area of packet radio networks and interconnection networks.

N. Mitton (✉)
Centre de recherche INRIA Lille - Nord Europe, Lille, France
e-mail: nathalie.mitton@inria.fr

Principles of geometric routing approaches are very simple. Every node is assumed to be aware of the location of itself, of its neighbors, and of the destination. Based only on these information, every node is able to perform a routing decision. The location can be determined by either geographic coordinates (we thus talk of geographic routing) or logical coordinates extracted from the environment. In the former case, location coordinates may be derived thanks to GPS or estimated thanks to any other positioning mean such as triangulation [2, 27]. In the latter case, a new coordinate system has to be built. Coordinates are said as *virtual coordinates*. Yet, geometric routing is

- localized: only local information such as the position of the current node holding a packet, the one of its neighbors, and the one of the destination is required to take a routing decision. Localized algorithms avoid communication overhead, which yields a scalable protocol;
- distributed: every node performs the same algorithm;
- memoryless: no additional information has to be stored neither on the nodes on the path nor in the message;
- scalable.

Indeed, unlike traditional routing schemes (either proactive or reactive), georouting does not need to flood the whole network and does not store any routing tables. These features make them more scalable (in terms of memory and bandwidth overhead) and more energy efficient (since no useless message is sent to discover routes).

Each of both families of georouting protocols (either with exact or with virtual coordinates) can be divided based on its properties with respect to the metric used (hop count or power) and whether or not it guarantees delivery. Therefore, there are four classes of algorithms: (i) simple hop count-based algorithms without guaranteed delivery, (ii) hop count based with guaranteed delivery, (iii) energy efficient without guaranteed delivery, or (iv) energy efficient and guaranteed delivery.

In this chapter, we review the different geometric routing protocols from the literature with respect to the kind of coordinates they use. Section 15.2 reviews the propositions based on geographic coordinates (latitude, longitude, altitude) while Sect. 15.3 focuses on protocols based on virtual coordinates. In each of these sections, we will see different approaches that have been proposed to achieve energy efficiency and packet delivery. In addition, Sect. 15.3 reviews several ways proposed to provide the virtual coordinate systems, highlighting strengths and weaknesses of each one. Finally, Sect. 15.4 summarizes the different solutions.

15.2 Geometric Routing Based on Geographic Coordinates

In this section, we consider that nodes are aware of their geographic coordinates. Like already mentioned, in such protocols, the routing decision is performed by a node holding a packet only based on the information of the position of itself, of its neighbors, and of the destination node. These positions may be exact if retrieved from a positioning device (GPS or Galileo). If only a part of nodes are equipped with a positioning device, geographical coordinates may be estimated through triangulation [27] or any other mean [6], based on the neighborhood tables and the coordinates of GPS nodes.

In order to cope with the lack of positioning information and take benefit from georouting paradigms, some works have proposed to evaluate the distance between each node in order to approximate the relative geographical position of the node in a two-dimensional plane. The approach proposed in [6] uses techniques such as RSSI (receive signal strength information), TOA (time of arrival), AOA (angle of arrival), or TDOA (time difference of arrival) [28] to estimate the distance between two nodes. Based on regular packet sending, every node knows its one-hop and two-hop neighborhood and knows some of the distances between its one-hop and two-hop neighbors. First, each node i chooses two nodes j and k from its one-hop neighbors that do not lie on the same line and with a known distance greater than 0. Node i then defines a local coordinate system based on i, j, and k. It holds coordinates $(0, 0)$. Using triangulation, each neighbor of node i can be positioned based on this local coordinate system. In a second phase, all local coordinate systems are modified by rotation or mirroring to achieve the same direction. In the third phase, an election algorithm is applied to choose the center of the network coordinate system and this network coordinate system is broadcast in the network. As a result, we obtain a relative and approximate (due to distance approximation) positioning system which is used exactly as coordinates provided by a GPS.

The position of the neighbors may be achieved through periodic exchange within a neighborhood where every node broadcasts its position in Hello packets. The position of the destination may be achieved through a localization algorithm like [6, 30, 40], but the review of localization scheme is beyond the scope of this chapter.

15.2.1 Greedy and Directional Approaches

First routing approaches were pretty simple and intuitive. For instance, in [26], the node holding a packet has to choose at random a forwarder among the neighbors in the forwarding direction of the destination. Yet, in Fig. 15.1, s will choose at random between nodes in the gray area, i.e., a, b, c, or e. Choosing the next hop only among the nodes in the forwarding direction guarantees that at each step, a progress is made toward the destination and no loop is created.

Then, in the greedy method [14], a node s holding a packet forwards it to its neighbor a that is the closest to the destination d. Greedy forwarding tries to bring the message closer to the destination at each step using only local information, aiming at reducing the overall number of hops. Thus, each node forwards the message to the neighbor that is most suitable from a local point of view. The most suitable neighbor in the "Greedy case" is the one who minimizes the distance to the destination at each step. In Fig. 15.1, node s sends its message to node a.

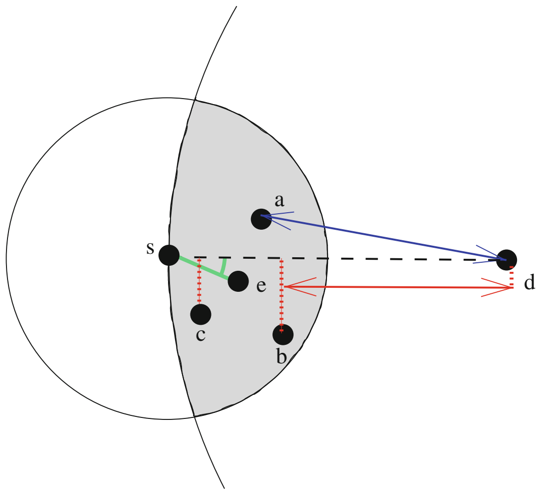


Fig. 15.1 Illustration of geographic greedy routings

Alternatively, one can consider another notion of progress. MFR [35] (most forward routing) and NFP [16] (nearest with forwarding progress) consider the projected distance on the source–destination line. In MFR, the sending node selects as the next forwarder the node with maximum projected distance while NFP selects the one with the smallest one. MFR tries to get closer to the destination at each step by sticking to the initial direction in order to limit the number of hops needed to reach the destination. NFP suggests to adjust the transmission power to the distance between the two nodes in order to save energy during the transmission. In Fig. 15.1, MFR would select node b while NFP would designate node c as the next forwarder.

Another alternative considers the neighbor which provides the minimum angle between source–neighbor and source–destination. This approach is known as DIrectional routing (DIR) or compass routing [18]. With such an algorithm, node a selects node e in Fig. 15.1. The idea here is to stick to the direct direction in order to reduce the stretch factor of the resulting routing path.

Basagni et al. incorporate mobility concerns by introducing DREAM [1]. Indeed, the source node first determines an angular sector for forwarding, based on the mobility information of the destination node d. Then, the message is forwarded to every node lying in that angular sector. To determine this angular sector, node s computes the circle centered on d with radius equal to the maximum possible movement of d since the last update. Then, the angular sector is defined by the tangent to that circle passing by node s. For instance, in Fig. 15.2, a forwards the message to every node in the gray area, i.e., nodes a and b. A very similar approach has been proposed in parallel than DREAM in the same conference. It is called LAR, for location-aided routing [17], and presents only few modifications. Here, if there is no node in the computed angular sector, this sector is enlarged till including one.

For all these methods, if the routing ends up at a node which has no neighbor closer than itself to the destination, the routing fails. In such a case, Finn [14] pro-

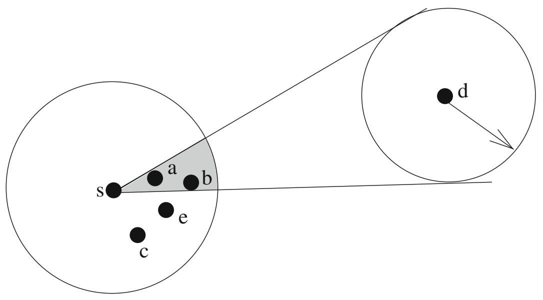


Fig. 15.2 Illustration of DREAM

poses to search all n-hop neighbors (nodes at distance at most n hops away from the current node, where n is a network-dependent parameter) by flooding the nodes until a node closer to destination than the current node is found. The algorithm has non-trivial details and does not guarantee delivery nor optimize flooding rate.

A variant of greedy algorithms, called GEDIR, is proposed in [33]. In this variant, the message is dropped if the best choice for a current node is to return the message to the node the message came from. It increases delivery rate by prolonging failure. The same criterion can be applied to MFR method and directional methods. GEDIR is often used as the basic ingredient in other routines. For instance, it is used in several location update schemes, such as quorum-based and home agent-based schemes.

Globally, basic greedy strategy based on distance is loop free. Indeed, at each step, the message has to move forward to the destination and thus cannot loop by going through a node it has already visited. MFR has also been proved to be loop free [33] while DIR is not loop free as shown in Fig. 15.3. Indeed, in Fig. 15.3, let us suppose that nodes a and b are not neighbors. By applying DIR, node s forwards to node a. c is the best choice for a as the next forwarder since it is the node which minimizes the angle toward d, which makes the message go backward. A loop then appears.

All the greedy approaches described to that point are based either on the distance or on the direction. None of them but NFP has energy consumption concerns. As already mentioned, NFP tries to minimize the energy consumption by sending the

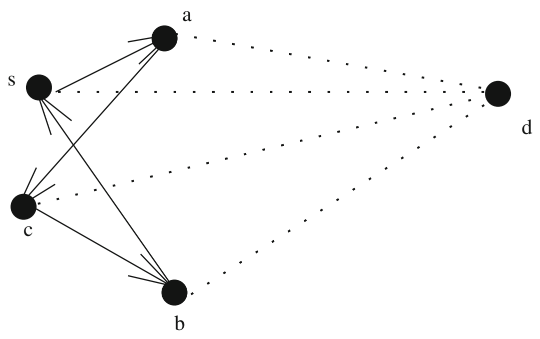


Fig. 15.3 DIR is not loop free

message to the closest node to a in the direction of the destination. This leads to a succession of small hops, less energy consuming than long hops. Nevertheless, generally, the energy consumed depends indeed not only on the transmission range r but also on the overhead c due to signal processing. The most commonly used energy model is

$$\mathrm{cost}(r) = \begin{cases} r^{\alpha} + c & \text{if } r \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{15.1}$$

where α is a real constant greater than 1 that represents the signal attenuation. Yet, a succession of small hops is not necessarily less energy consuming than a succession of a smaller number of greater hops.

In [34] the optimal transmission radius, r^*, that minimizes the total power consumption for a routing task is computed and it is equal to

$$r^* = \sqrt[\alpha]{\frac{c}{\alpha - 1}} \tag{15.2}$$

Based on these observations, the first power-aware localized routing algorithms were described in [34]. Cost-over-progress (COP) framework with power as the cost has been applied in [19]. Let us take Fig. 15.4 to illustrate it. To forward a packet to destination node d, source node s considers only nodes in the forwarding direction of d (nodes in the gray area). It selects among them its neighbor a such that the ratio of the energy consumed to reach that neighbor (cost($|sa|$)) to the progress made (measured as the reduction in distance to d, i.e., $|sd| - |ad|$) is minimized. The idea is the following. Ideally all hops from nodes s to d provide the same progress as the first one via candidate neighbor a. The number of such hops along the path from s to d is then $|sd|/|sd| - |ad|)|$, and cost of each is cost($|sa|$). $|sd|$ being a constant, at each step, the algorithm tries to optimize the ratio cost($|sa|)/|sd| - |ad|$.

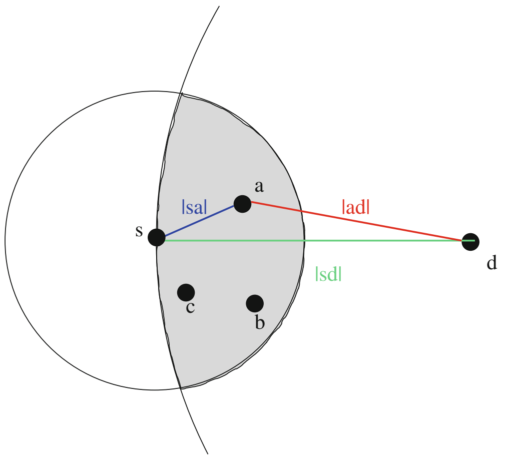


Fig. 15.4 Illustration of COP

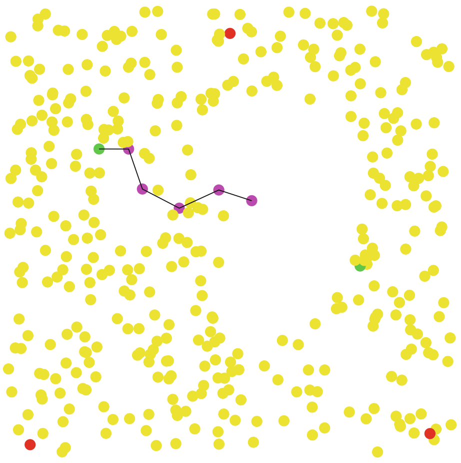


Fig. 15.5 Illustration of a failure routing. Source and destination nodes are the green nodes. Nodes on the routing path appear in *red*. Routing reaches a coverage hole and fails

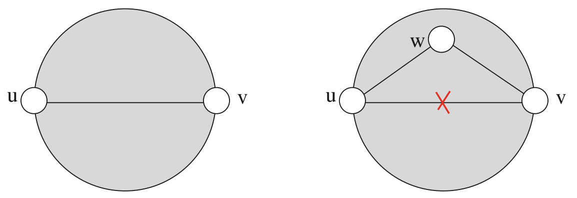
The same paper [19] proposes and analyzes another algorithm. The iterative power progress algorithm is an improvement of the basic COP algorithm. It works as follows. As in COP, a node s, currently holding a message destined to d, first finds a neighbor a that minimizes $\text{cost}(|sa|)/(|ds| - |da|)$. Then, the search continues for an intermediate node b which (i) is closer to d than s, (ii) is neighbor to both s and a, and (iii) satisfies $\text{cost}(|sb|) + \text{cost}(|ba|) < \text{cost}(|sa|)$ and has the minimum $\text{cost}(|sb|) + \text{cost}(|ba|)$ measure. If found, such node b replaces a as selected neighbor, and the search for a better intermediate node repeats. This process is iteratively repeated until no improvement is possible, and node s forwards the message to the selected neighbor, which then applies the same scheme for its own forwarding.

Till that point, we have reviewed most of the greedy routing protocols, hop count based or energy aware. Though these routing algorithms work well in dense networks, they fail if the node holding the message is closer to the destination than any of its neighbors. Indeed, in sparse graphs, these algorithms suffer from coverage hole and may fail, as Fig. 15.5 illustrates. Therefore, some studies introduce solutions that guarantee delivery.

Yet, several propositions have been proposed in the literature for greedy routing in wireless networks, with energy concern when the radius range can be adapted on demand by nodes. Nevertheless, none of these approaches guarantee the packet delivery even if the network is connected. Therefore, more investigations have been performed on this track.

15.2.2 Guaranteed Delivery Approaches

In order to guarantee the message delivery, the authors of [4] have proposed the face routing. Face routing guarantees delivery in two-dimensional UDG. Face routing requires the network topology to be a planar graph (i.e., no edges intersect each other). To planarize a graph, several algorithms can be used such as Gabriel



(a) Case edge *uv* belongs to GG.

(b) Case edge *uv* does not belong to GG.

Fig. 15.6 Illustration of Gabriel graph building. In (**a**), there is no node lying in the disc with diameter $[u, v]$, edge uv belongs to the GG. But, in (**b**), the presence of node w in the disc removes edge uv from GG

graph(GG) [4, 20], the relative neighborhood graph (RNG) [36], or the Morelia graph [3]. Gabriel graph, for instance, contains edges between nodes u and v if and only if no other nodes are located inside the circle of diameter $|uv|$, as illustrated in Fig. 15.6. GG has some desirable properties when used for routing in wireless networks such as localized message, free computation, planarity, and preserving connectivity [4]. If the initial underlying graph is a graph $G = (V, E)$, the outcome planarized graph is a subgraph $G' = (V, E') \subset G$ where every intersecting edge has been removed.

Planarization divides the network into faces. Face routing then performs as follows on the planarized graph G'. The face that contains the line (sd), where s is the source node and d is the destination node, is traversed by right-hand or left-hand rule (placing a virtual hand on the wall of the face). When edge which has to be followed intersects with the imaginary dash line (sd), the message changes face, and so on, till reaching the destination node. Figure 15.7 illustrates the face algorithm. To send a message to node d, s follows the first face (composed by nodes $sabgh$) and forwards it to node a. a applies the same algorithm and forwards to b. If b continued to follow the same face, it would send the message to node g and by doing so would cross red line (sd). Therefore, b changes face and sends the message to node c. The same algorithm is applied by every node on the path. Finally, the message follows blue edges in figure, going through nodes $sabciefd$.

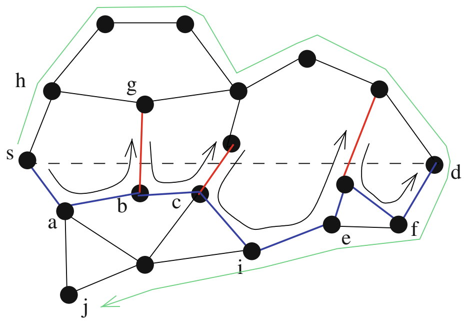


Fig. 15.7 Illustration of face routing. Path from s to d appears in *blue*, path from s to j in *green*

Theorem 1 *FACE guarantees the packet delivery.*

Proof We give a sketch of proof of a variation of the FACE routing algorithm. This version of FACE is the following. We consider the current node s with the destination node d. At each step, the current node s knows the position of the destination d and has in input a reference point r which is used to decide the face traversal and the change of face. At the starting point, we take $r = s$. The current node s selects the face which contains the segment $[rd]$, and the message starts the face traversal according to the right-hand or left-hand principle. The change of face is decided when the message crosses the segment $[rd]$ by using an edge (b, g) (by using Fig. 15.7 notations). In this case, we set $r =]rd] \cap [bg]$ and $s = b$. We can note that it has been shown that we can also apply $s = g$ without real impact on the correctness of the algorithm or on the performances of the routing protocol. After changing face, we apply the same algorithm in order to select the next face.

If at each face traversal we find an edge (b, g) which crosses $]rd]$, we can observe easily that the distance $|rd|$ is strictly decreasing and then the delivery is guaranteed. Then, without loss of generality, it is enough to show that the edge $]rd]$ is always crossed in the face traversal. This is equivalent to show that there always exists an intersection between the segment $]rd]$ and the selected face.

We distinguish two cases: the selected face is an internal face or the selected face is an external face of the graph of the whole network. In the case of internal face, the crossing of the segment is guaranteed if the destination node is not isolated in the middle of the selected face since the selected face contains the segment. This case cannot happen if the network is connected. In the case of external face, if the destination node is not part of the external face, the existence of an intersection between the segment and the face is sure since a part of the external face is situated under the $]rd]$ segment and another part is situated above the segment. Indeed, the segment $]rd]$ "enters" in the graph at a point A which is at the intersection of the face and the segment. If the destination node D is part of the external face, the existence of the intersection is trivial. ∎

Yet, face routing guarantees message delivery. But using only face routing may generate very long paths in cases where the message has to follow the external face of the network. This is, for instance, the case in Fig. 15.7 when node s needs to reach node j. The message follows the green line. Therefore, to overcome this drawback and to take advantages of both greedy and face solutions, the authors of [4] propose the GFG (greedy–face–greedy)-based approach. It applies greedy routing until either the message is delivered or the routing fails. In the latter case, face routing is applied to recover from failure. It has been shown in [15] that face routing guarantees recovery traversing the first face. Greedy routing continues from a until delivery or another failure node is encountered. Through guaranteed delivery and providing path with a fairer stretch factor than face, GFG remains not energy efficient.

The first tentative to address guaranteed delivery in power-aware localized routing is [32]. It is a greedy–face–greedy (GFG) approach where greedy routing is the COP as in [34] while face routing is similar to the one in [4]. One of the drawbacks of face routing is that it is likely to follow a long sequence of short edges of GG. Although short edges are less energy consuming than long edges, a succession of short edges will be more energy consuming than a short sequence of medium edges. Yet, for sparse networks where the face step is often triggered, this approach is not energy efficient.

In LEARN [39], a localized energy-aware routing is proposed. LEARN assumes that every node is aware to adapt its transmission range. A node s aiming at destination d selects neighbor b inside a restricted neighborhood ($\widehat{bsd} \leq \alpha$ for a parameter $\alpha < \pi/3$) that has the largest energy mileage, determined as the ratio $|sb|/\text{power}(|sb|)$ where $\text{power}(|sb|)$ represents the cost to send a message from node s to node b. If no such neighbor exists inside the restricted neighborhood, LEARN fails. In the variant LEARN-G, a node switches to greedy routing [14] in case of failure and selects the neighbor closest to the destination. Finally, in the variant LEARN-GFG, a node invokes face routing when a failure occurs. Thus, as previously, LEARN can be energy efficient only when the network is dense enough and that every node on the path may find a neighbor of it in the α angular sector toward the destination (no invocation to greedy or face procedures).

The authors of [39] show that when LEARN indeed finds a path without needing to invoke neither greedy nor face routing, the total Euclidean length of the final path from s to d is within a constant factor of the optimum direct path (see Theorem 2) and that this path is energy efficient. These proofs have then been generalized and extended in [12] to any protocol, providing paths meeting angular constraints as follows:

Definition 1 A path meets angular constraints if every hop is within an angle $\theta \leq \alpha < \frac{\pi}{3}$, $\theta \to 0$ toward the destination.

Theorem 2 (Wang et al. [39]) *Any path from s to d meeting angular constraint has length that is a constant of the optimum $|sd|$.*

Proof The proof is made by induction on the number of hops. The theorem is clearly true when the path has only one hop (the path has optimal length). Assume that it is true for the path with $(k-1)$ hops. Then consider any k-hop path $v_0v_1v_2\ldots v_{k-1}v_k$. By induction, the length of path $v_0v_1\ldots v_{k-1}$ is at most a constant δ from the optimum: $\sum_{i=1}^{k-1}|v_{i-1}v_i| = \delta \times |v_0v_{k-1}|$. Then it is sufficient to show that $|v_{k-1}v_k| + \delta \times |v_0v_{k-1}| \leq \delta \times |v_0v_k|$.

Let us consider triangle $v_0v_1v_k$, in which, by the routing protocol, v_0 is the longest link and $\angle v_1v_0v_k = \theta < \alpha$ (angular constraints). Let us note $\angle v_0v_kv_1 = 2x$. Then a simple geometry computation shows that $\frac{|v_0v_1|}{|v_kv_0|-|v_kv_1|} = \frac{\sin(x+\frac{\pi}{2})}{\frac{\pi}{2}-x-\theta} = \frac{\cos x}{\cos(x+\theta)}$. This means that we need $x < \frac{\pi}{2} - \theta$ and $x < \frac{\pi-\theta}{4}$. Simple computation shows that $\frac{\cos x}{\cos(x+\theta)} < \frac{1}{1-2\sin\frac{\theta}{2}} < \frac{1}{1-2\sin\frac{\alpha}{2}}$ for $x < \min\left(\frac{\pi}{2}-\theta, \frac{\pi-\theta}{4}\right)$. And thus $|v_{k-1}v_k| + \delta \times |v_0v_{k-1}| \leq \delta \times |v_0v_k|$ for every routing protocol respecting angular constraints. ■

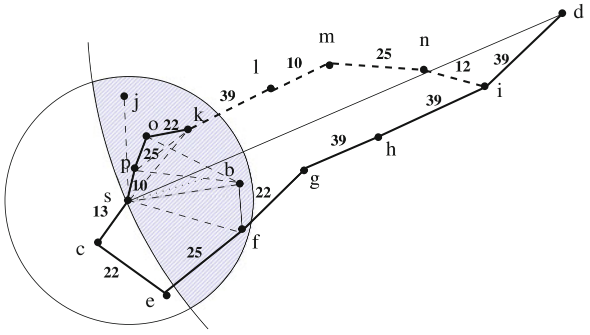
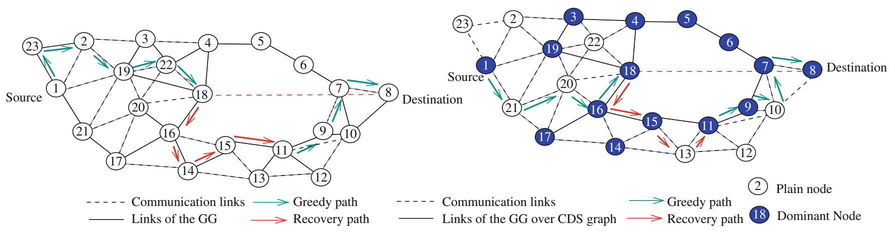


Fig. 15.8 Illustration of the greedy step of the SPFSP and EtE. *Dash links* are the links with high weights

In [31], SPFSP (shortest path face shortest path), a GFG-based energy-aware routing with guaranteed delivery, has been proposed. The energy awareness is introduced via the computation of an energy-weighted shortest path (with power consumption as weights) at both greedy and face steps. Let us take the example plotted in Fig. 15.8 to illustrate this greedy step. Node s currently holding a packet first selects a target node by using the plain greedy algorithm [14], i.e., its closest neighbor to destination d: node b in figure. Instead of transmitting directly to b, s computes the energy-weighted shortest path to b over its whole neighborhood. In the figure, this path is $scefb$. Indeed, node s also considers its neighbors that are not in the forwarding direction of d like nodes c and e. This path is then followed by the packet until reaching node f, which is the first node on the path closer to d than s. Node f then recursively applies the same protocol till either reaching the destination node d or to fail. Note that, to ensure no loop, the shortest path has to be embedded in the message, creating an overhead. To recover from failure points, a face routing [4] is used in the following way. If we suppose that node s in Fig. 15.7 is the failure node, it applies face routing only to determine the target node: node a in Fig. 15.7. But, once again, instead of reaching that target node directly, node s computes an energy-weighted shortest path over its neighborhood and reaches node a via this shortest path. Nevertheless, simulations have shown that most of the time, this enhancement added to the recovery step is of no use since the shortest path is most of the time the edge of the face itself. This is due to the fact that GG keeps only small edges in the underlying graph and, thus, face edges are among the smallest ones and thus among the less consuming edges within a node neighborhood.

Figure 15.9a illustrates a sample execution of the SPFSP algorithm. Greedy routing proceeds from node 1 which first chooses its next forwarder among nodes 2, 19, and 21 (node 23 is not included in the selection since it is further from the destination than node 1). Node 1 selects node 19 as temporary destination since it provides the best progress toward destination and sends the packet to node 23, the first node on the shortest path toward node 19. Note that even if node 23 was not among the potential targets, it may be included in the path to it. Node 23 then forwards to node 2, second node on the path embedded in the packet by the source node. Node 2 being closer to node 8 than the source, it performs the selection algorithm and finds node 19 as its best forwarder and in this case, the shortest path in that link. Node



(a) Illustration of SPFSP. (b) Illustration of EtE.

Fig. 15.9 Comparison of SPFSP and EtE. Continuous links are the links to be followed in the recovery step (GG). GG is built over every node in plain face routing or in SPFSP (**a**), while it is computed only over dominant nodes (*blue nodes*) in EtE (**b**)

19 selects node 18 by following a shortest path through node 22 to which it sends the message. The latter then forwards to node 18 where greedy routing fails. Face routing is then invoked to follow edges 18-16 (directly), 16-14, 14-15, and 15-11. Greedy routing then continues till final delivery.

Based on these observations, the authors of [12] have proposed EtE (end-to-end) protocol which guarantees the packet delivery with energy concerns at both greedy and face steps. EtE draws its inspiration from SPFSP. The greedy step is modified in two ways: (i) in the way the selected target is chosen and (ii) on the set of nodes over which the shortest path is computed. Indeed, in order to avoid to embed the path in the packet, the shortest path is computed only on nodes in the forwarding direction of the destination d. In Fig. 15.8, s computes the shortest path only over nodes in the blue dash area, i.e., b, f, j, k, o, and p. The selection of the target node is modified as follows. Instead of selecting the closest node to the destination, node s selects its target node in a *cost-over-progress* fashion [19] where the cost is the cost of the energy-weighted path from node s to the considered node u. Let $v_0v_1...v_iv_{i+1}..v_n$ be the nodes on the shortest path from s to k with $v_0 = s$ and $v_n = u$. The cost of the shortest path $\text{cost}_{\text{SP}}(s, u)$ from s to u is defined as

$$\text{cost}_{\text{SP}}(s, u) = \sum_{i=0}^{n-1} \text{cost}(|v_i v_{i+1}|)$$

Then, node s selects node b which minimizes the cost of the shortest path from s to u divided by the progress it makes toward destination node d.

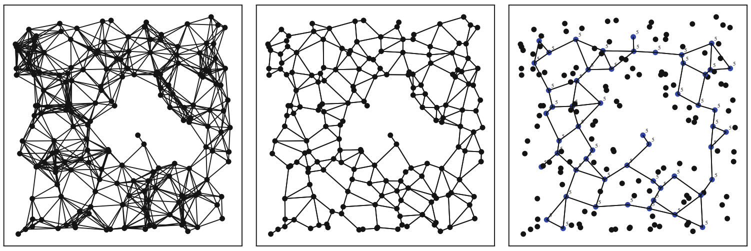
The target node k is then the one which is such that

$$\frac{\text{cost}_{\text{SP}}(s, k)}{||sd| - |kd||} = \min_{u \in N_d(s)} \frac{\text{cost}_{\text{SP}}(s, u)}{||sd| - |ud||}$$

Once that node is determined, node s forwards the message to the first node on the shortest path from s to k, i.e., node p in figure. Then, node p reiterates the same process. No loop is possible since the next hop is computed only over nodes in the forwarding direction of the destination. This algorithm goes on till the destination is reached or there is no node closer to the destination than the one currently holding the message. In this latter case, an energy-efficient face routing is applied for recovering from failure.

As already mentioned, regular face routing guarantees delivery but is not energy efficient since it may use very short edges compared to the energy-optimal transmission length r^* since GG removes longer edges. To overcome this drawback, the authors of EtE have introduced an energy-efficient variant to face routing. For it, they add a step in the planarization of the graph. From the original graph $G = (V, E)$ (Fig. 15.10a) they compute a connected dominating set (CDS), V' of V. Since source s and destination d may not be in the CDS, the set V' is expanded with s and d. Let $G' = (V', E') \subset G$ where $V' \subset V$ is the expanded set of dominant nodes and $E' \subset E$ is the set of edges between nodes in V'. Then, a CDS election protocol is applied on V'. The authors use the CDS election protocol introduced in [5], but any other election protocol may be applied. Since face routing must be applied on a planarized graph, the Gabriel graph $G'' = (V', E'')$ is extracted from G', where $E'' \subset E'$ is the set of edges remaining in the planarized graph (Fig. 15.10c). Face routing is lastly run over G''. This face routing guarantees delivery in the constructed subset since it contains source and destination nodes and preserves connectivity. Moreover, by considering only edges connecting two dominating nodes, the routing process avoids to choose too short edges. Each node needs to know its neighbors that are in the CDS.

Based on G'', the same principles of the recovery step of [31] are applied. Current node s that is in recovery mode applies face routing on G'' only to decide on which edge (s, b) to follow to reach the destination node from a given node s. However, that edge does not need to be selected since it may be too long ($|sb| > r^*$). Node b is thus reached through an energy-weighted shortest path. If b is closer to the destination node than the node which has initiated the face routing step, node b selects the next hop in the routing path following the greedy routing described above. Otherwise, it determines the node following face routing over CDS nodes and computes the energy-based shortest path to reach it.



(a) Initial graph G (b) GG on G (c) GG from G'

Fig. 15.10 When using regular face routing over graph G (**a**) (like SPFSP), messages follow GG edges of G (**b**). Edges of G'' (**c**) are used instead when applying the energy-efficient face routing of EtE

Figure 15.9b illustrates a sample execution of EtE algorithm. Greedy routing proceeds from node 1 which first computes the cost of the shortest path toward nodes 2, 19, and 21. Node 1 selects node 19 as temporary destination since it provides the lowest cost over progress and sends the packet to node 21, the first node on the shortest path toward 19. (Node 23 is not included in the computation since it does not provide any progress to destination.) Node 21 finds node 20 as its best forwarder and in this case, the shortest path is that link. Node 20 selects node 18 by following a shortest path through node 16 to which it sends the message. The latter then forwards to node 18 where greedy routing fails. Face routing is then invoked to follow edges 18-16 (directly), 16-15 (directly), and 15-11 (which is replaced by path 15-13-11 for energy efficiency). Greedy routing then continues till delivery to 11 selecting 10 via 9, 9 selecting 10, and 10 selecting destination 8 and delivering via node 7.

Yet, we have reviewed most of the geographic routing for wireless sensor and ad hoc networks in the literature. Even if along the years the solutions proposed are better and better, the ones of them that guarantees delivery all assume a unit disc graph. Indeed, if this assumption does not hold, there is no way to planarize the graph, and thus face routing and variants cannot be applied anymore. Unfortunately, last experiments have shown that the UDG does not hold in a realistic network. Yet, the new challenge appearing here is to investigate geographic routing solutions that both are energy efficient and guarantee delivery but in any arbitrary graph (not only unit disk graph).

15.2.3 Anycasting

In the anycasting problem, a sensor wants to report event information to one of sinks or actors. The authors of [25] describe the first localized anycasting algorithms that guarantee delivery for connected multi-sink sensor and sensor–actor networks.

Three geographic anycast algorithms are proposed: GFGA, COPA, and EEGDA which are inspired, respectively, by GFG [4], COP [34], and EtE [12]. Each of them consists of greedy and recovery phases, and they all guarantee delivery for a report from a sensor if it is connected to at least one sink or actor. GFGA uses hop count as the metric, while others apply power consumption, where both greedy and recovery steps are energy efficient. The two energy-efficient algorithms have different computing complexities.

All algorithms construct a path from the source sensor node to one of sinks/actors. During the path construction, there exists a single destination to reach. The main feature of these algorithms is that this destination may change along the path of the message according to the network topology. Anycasting may start from current node s toward sink/actor $S(s)$ that is the closest to it. However, s could in fact be even disconnected from $S(s)$ or closer in number of hops to another sink.

In greedy phase, and with hop count as the metric, i.e., GFGA, node s currently holding the packet forwards it to its neighbor v that minimizes $|vS(v)|$ (is closest to its nearest actor/sink). When an arbitrary cost metric is used (COPA), the selected neighbor v minimizes the ratio of cost cost($|sv|$) of sending packet to v over the reduction in distance ($|sS(s)| - |vS(v)|$) to the closest actor/sink. An improved variant, EEGDA, is to forward to the first neighbor on the shortest weighted path toward v instead of sending directly to v, like in EtE. If none of neighbors reduces that distance then recovery mode is invoked. It is done by face traversal toward the nearest connected actor/sink, where edges are replaced by paths optimizing given cost. A hop count-based (FACE-like) and two variants of localized power-aware (Ete-recovery-like) anycasting algorithms are described.

15.3 Virtual Coordinate Systems

As noticed in previous sections, georouting solutions are very promising solutions for wireless sensor networks since coordinates simplify the routing decision at each node by limiting bandwidth and memory overheads. Nevertheless, geographical position information provided by devices such as GPS or Galileo is not always a feasible solution. Indeed GPS-like positioning systems are bulky, energy costly, and expensive and are not adapted for every environment [28]. Therefore, literature has witnessed the birth of protocols that assign "virtual" coordinates to each node to take benefit from georouting techniques.

Indeed, it is worth noting that virtual coordinates do not necessarily need to embed global positioning information, and they just have to be consistent enough to allow georouting. Two well-spread methods have been proposed. The most used positioning technique is based on hop count distance from given landmarks and is described in Sect. 15.3.1. Many routing protocols are based on this virtual coordinate system. They mainly differ in the distance functions and routing progress they use, as we will see. Though simple, such a landmark-based system exhibits some drawbacks due to coordinate constructions that we will detail. Therefore, a new tree-based coordinate system has been proposed (Sect. 15.3.2).

15.3.1 Landmark-Based Coordinate System

Landmark-based coordinate system is based on hop distance between the sensors and some specific nodes and do not try to approximate physical coordinates. Therefore, the virtual topology can be unrelated to the physical topology of the network.

15.3.1.1 Landmark-Based Coordinate System Construction

The landmark-based coordinate system is built into two steps: (i) flooding by landmarks and (ii) computation of coordinates. The first step is common to every landmark-based georouting protocol while the second one differs from one to another.

The first step can be split into two phases. In the first phase, a global and distributed election mechanism elects a set of nodes as landmarks or anchors. Nodes acting as landmarks can be explicitly designated by an external process at the bootstrap of the network. During the second phase, every landmark floods a message containing a counter which is incremented at each hop. In the sequel, the term "broadcast" stands for message propagation in a node's neighborhood and the term "flooding" refers to network-wide message propagation. At the end of this second phase, every arbitrary node i can thus determine a vector $V(i) = \left(h_1^i, ..., h_n^i\right)$ where n is the number of landmarks and h_n is the hop distance between node i and each anchor node (node i is h_1^i hops away from landmark 1).

The second step allows every node i to compute its virtual coordinates based on vector $V(i) = \left(h_1^i, ..., h_n^i\right)$. To do so, different functions can be used depending on the protocol. The virtual coordinates of node i are $X(i) = \Gamma(V(i))$ where $X(i) = \left(x_1^i, ..., x_m^i\right)$ and $m \leq n$ where different Γ functions are used in the literature. The most common Γ functions are the following ones:

- The "identity" function denoted by Γ_{id} where $X(i) = V(i)$ with $m = n$. This is the simplest function used, for instance, in VCap [2, 7] or VCost [11].
- The "centered virtual coordinates" function [13] denoted by Γ_{cvc} and $x_j^i = \left(h_j^i\right)^2 - \mu$ for $m = n$, $j = 1, \ldots, n$ where $\mu = \frac{1}{n} \sum_{j=1}^{n} \left(h_j^i\right)^2$.
- The "averaging" function [23] denoted by Γ_{av} gives the following relationship between $X(i)$ and $V(i)$:

$$x_k^i = \frac{\frac{\sum_{t=1}^{|N(i)|} h_k^t}{|N(i)|} + h_k^i}{2}$$

where $|N(i)|$ is the number of neighbors of node i.

All these Γ functions are used to compute the virtual coordinates of nodes based on their hop distance to every landmark. As next section will show, they all exhibit different features and impact the routing process. Figure 15.11 illustrates the results of coordinate assignment with identity Γ function (Γ_{id}) such as in VCap [7] and/or JUMPS [2]. In this figure, landmarks (anchors) are arbitrarily chosen as nodes L_1, L_2, and L_3. Therefore, every node has a three-dimensional vector as coordinates constituted by the number of hops between itself and every landmark. In this figure, landmark L_3 has coordinates (4, 3, 0) since it is four hops away from landmark L_1 (through nodes 2, 3, and 11), three hops away from landmark L_2 (through nodes 7 and 11), and zero hop away from itself. Since flooding message from each landmark may be received more than once, a node chooses the minimum hop distance to each landmark.

Georouting is then performed on top of $X(i)$ coordinates. In greedy methods, as in geographic routing based on real coordinates, the next hop is the node u that minimizes the distance $\delta(u, d)$ to the destination d (maximal progress). Distance $\delta(u, d)$ is computed over virtual coordinates and once again, several distance functions can be used according to the routing protocol. The most common distances found in the literature are the following:

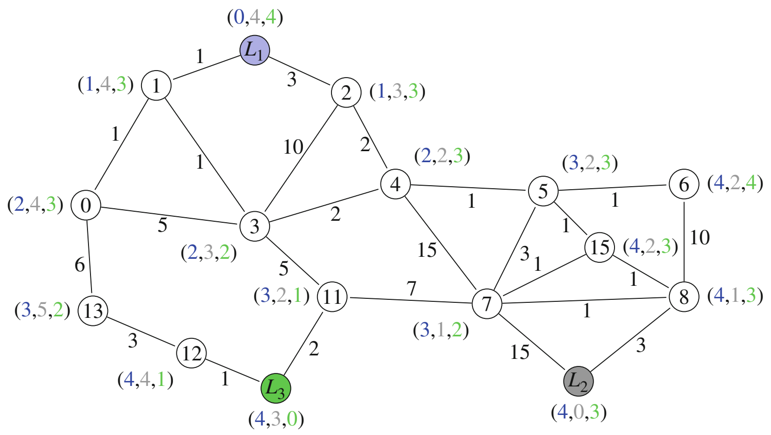


Fig. 15.11 Illustration of hop distance coordinate assignment. In this figure, L_1, L_2, and L_3 are the landmarks. Each node, including landmarks, has a three-dimensional vector as coordinates

- Euclidean distance $\delta_e(u, d) = \sqrt{\sum_{j=1}^{n} \left(x_j^u - x_j^d\right)^2}$,
- Hamming distance $\delta_h(u, d) = \sum_{j=1}^{n} \left|x_j^u - x_j^d\right|$,
- Square Euclidean distance $\delta_s(u, d) = \delta_e(u, d)^2$.

It is worth noting that combining the different Γ and δ functions provides a great amount of different techniques. In the next section, we review the protocols from the literature, explaining what combination they use.

15.3.1.2 Routing on Top of Landmark-Based Coordinates

Even if routing based on landmarks coordinate system may not always be successful, there exists many georouting protocols based on hop distance but their performances differ due to the distance (δ function) and coordinate (Γ function) computation used in each protocol. This section reviews and classifies georouting based on landmark-based system.

The simplest algorithm is VCap. In *Vcap* [7], greedy georouting is performed on top of landmark-based coordinates by combining the Γ_{id} function to compute node coordinates and the δ_e distance function. The system is composed of three landmarks. The next hop is the node u that minimizes the Euclidean distance $\delta_e(u, d)$ to the destination d. $\delta_e(u, d)$ is computed over virtual coordinates as follows: $\delta_e(u, d) = \sqrt{\sum_{j=0}^{n} \left(h_j^u - h_j^d\right)^2}$. To illustrate a routing process, let us assume that node L_2 needs to send a packet to node L_3. To select the next hop, node L_2 computes the distance from all its neighbors to the destination. If the Euclidean distance is used, $\delta_e(7, L_3) = \sqrt(9)$, $\delta_e(8, L_3) = \sqrt{13}$, and $\delta_e(L_2, L_3) = \sqrt{18}$. L_2 chooses among its neighbors u with positive progress ($\delta(u, L_3) < \delta(L_1, L_3)$), the closest one to the destination. Since $\delta_e(7, L_3) < \delta_e(8, L_3)$, L_2 chooses node 7 as its next hop. Then, node 7 elects node 11 since $\delta_e(11, L_3) < \delta_e(15, L_3) < \delta_e(5, L_3) < \delta_e(4, L_3) < \delta_e(L_1, L_3)$. As node 11 is a neighbor of node L_3, the greedy routing technique is successful.

JUMPS [2] provides a coordinate system similar to VCap. JUMPS only differs from VCap in the fact that it may use more than three anchor nodes. Similar to VCap, Γ_{id} is chosen as the basis of the coordinate system. Obviously, using these kinds of coordinates may end up to several nodes holding the same virtual coordinates leading to routing ties, which reduces the delivery ratio. Therefore, in addition, JUMPS provides a study on the impact of the landmark placement on routing delivery rate. It happens that the better landmark placement is when landmarks are spread at equal distance one from each other all around the network. This is indeed the placement which reduces the number of nodes holding the same coordinate.

To palliate or at least reduce this drawback and thus increase the delivery rate, the authors of *AVCS* [23] do not use necessarily integer coordinates. Floating coordinates depending on the neighborhood of each node are used instead of hop count. AVCS uses the Γ_{av} which performs a centroid transformation as an averaging function to compute the floating coordinates of the node. The authors of AVCS use the Euclidean distance but with different virtual coordinates. Indeed, the distance between the current node i and the destination node v is $\delta_e(V(v), X(i))$. Results show that a greedy routing on top of AVCS coordinate system outperforms greedy routing protocol on top of geographical coordinates since using virtual coordinates avoids the routing holes. It is worth noting that in AVCS, each node keeps the $V(i)$ and $X(i)$ coordinates which increases memory consumption. The authors also suggest the possibility of applying the Γ_{av} function more than once to reduce the probability of having the same coordinates for two nodes. That is, the Γ_{av} function is applied on $X(i)$ coordinates. Moreover, and in order to even more reduce redundant coordinates, the authors suggest to apply the Γ_{av} function by using the two-hop neighborhood of a node.

In *Gliders* [13] another way of assigning virtual coordinates is described to avoid bad placement of landmarks by taking into account holes in the network. Nodes are partitioned into tiles and landmarks are selected using Voronoi cells [38]; combinatorial Delaunay triangulation is used to estimate the global topology. Virtual coordinates for each node are derived from the node's distance (hop distance) to nearby landmarks by using Γ_{CVC} function. The authors of [13] also describe a routing protocol associated with this coordinate system. In their routing protocol, nodes have to compute a sequence of tiles for inter-tile routing paths, then a gradient descent procedure based on a proper distance function close to the Euclidean distance is used to route packets in a greedy way for intra-tile routing. However, if the coordinate system proposed in [13] can avoid routing ties depending on the density, the routing protocol does not guarantee packet delivery since packet may still reach dead ends. In addition, such a coordinate system associated with such a routing algorithm are very complex and induce a huge memory, bandwidth, and computational overhead, which makes it not scalable and difficultly implemented.

15.3.1.3 Energy Efficiency

In order to increase the delivery rate of georouting protocols using landmark-based coordinates, the authors of *VCost* [11] explore the use of several kinds of Γ and δ functions for coordinates construction and compare their performance. Moreover, they suggest the use of the Hamming distance δ_h instead of using Euclidean distance for routing decision. Results show that the use of Hamming distance increases the delivery rate and can reduce the path length compared to Euclidean distance since Euclidean distance first tries to minimize the maximum difference between coordinates (see Sect. 15.3.1.4). But the main goal of the work presented in [11] is first to provide an energy-efficient georouting on top of virtual coordinates. Therefore, the authors of [11] evaluate and reduce the energy consumption by using a cost-over-progress fashion (see Sect. 15.2.1) to reach the destination on top of virtual coordinates achieved through the Γ_{id} function. Authors assume that nodes are able to tune their transmission range and to estimate the cost of a transmission to each of their neighbors. Nodes select their following next forwarder as the node which minimizes the ratio between the cost of the transmission to the progress provided by this neighbor. The progress is thus computed as the Hamming distance between considered neighbor and destination node. VCost is the first power-aware georouting on top of virtual coordinates.

To illustrate the routing decision of VCost, let us consider the network depicted in Fig. 15.11 and assume a routing from node 4 to node L_2. The cost of each link is given on each edge, and we can see that the cost of the link between node 4 and node 5 is 1. Therefore, we have $\delta_h(L_2, 4) = 4$, $\delta_h(L_2, 5) = 3$, and $\delta_h(L_2, 7) = 3$. The progresses are equal for node 5 and node 7. However, the cost of each link is different, and node 5 is chosen as the next hop of the routing process. The path from node 4 to node L_2 uses nodes 5, 15, and 8.

15.3.1.4 Landmark-Based Positioning System Issues

Yet, routing performed over landmark-based positioning system exhibits some interesting properties. Nevertheless, landmark-based coordinate systems show some issues due to its construction. This section illustrates such particular issues which may reduce the performance of the routing protocol.

Dead End

Let us first consider a packet from node 6 to node 15 in Fig. 15.11. In this case, the greedy technique does not succeed since $\delta_e(6, 15) < \delta_e(15, 5) = \delta_e(15, 8)$ and thus, there is no neighbor of node 6 closest than itself to node 15. This leads to a dead end at node 6. This can arise when using the Γ_{id} function to compute coordinates, but such a situation may also be reached when using other Γ functions. The choice of this function will only impact the number of such situations.

But dead ends can also occur because of another configuration illustrated by Fig. 15.12a. In this figure, node 0 wants to send a packet to node 4. Since, $\delta_e(5, 4) <$

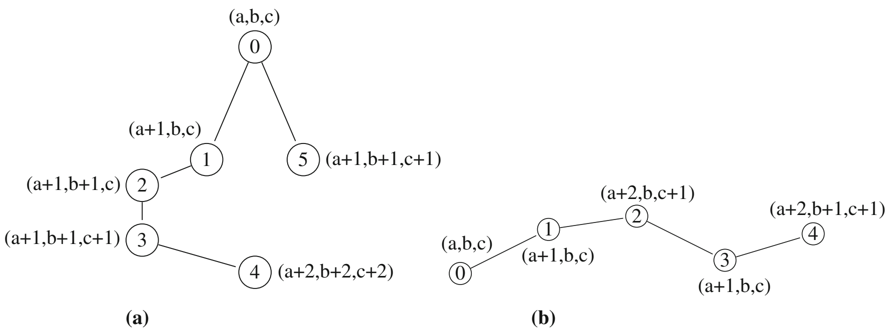


Fig. 15.12 Dead end: (**a**) due to holes and (**b**) due to duplication

$\delta_e(1,4) < \delta_e(0,4)$, node 5 is chosen as the next hop, which leads to a dead end. This latter case is actually the same as the one encountered with greedy algorithms based on geographical coordinates.

Duplicated Addresses

Landmark-based coordinate system may lead to multiple nodes holding the same coordinates. This duplication may have no effect since nodes with the same coordinates are geographically close to each other. However, this is not always the case. Duplication may also lead to dead end since coordinate uniqueness is mandatory to ensure the packet delivery. In Fig. 15.12b, node 3 and node 1 have the same coordinates, therefore, a packet transiting from node 0 aiming to node 4 will go through node 1 and then will stop at node 2 since $\delta_e(2,4) < \delta_e(3,4)$.

Duplicated addresses may also lead to routing ties. Figure 15.13a shows how it can happen. In this figure, node 0 and node 1 hold the same coordinates while they can be geographically far from each other. This duplication is due to the alignment

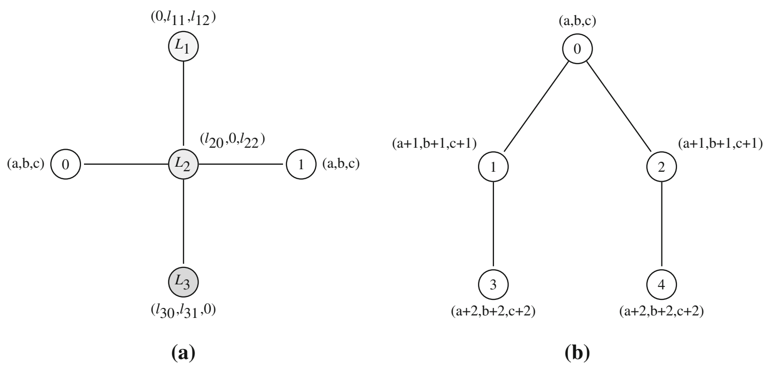


Fig. 15.13 Routing ties: (**a**) bad placement of landmarks and (**b**) one-connected networks

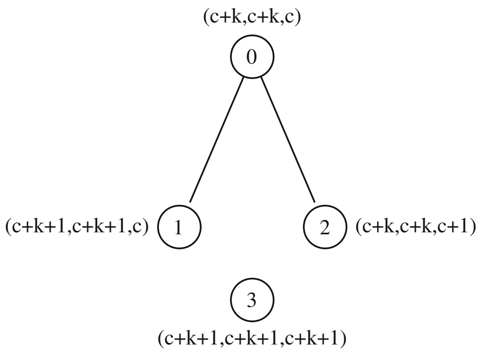


Fig. 15.14 Euclidean versus Hamming distance

of landmarks, which may lead to routing ties. In Fig. 15.13b a part of the network is one-connected, which may lead to the same effect.

Euclidean Versus Hamming Distance

Figure 15.14 shows an example where Hamming distance can reduce the number of hops. In this figure, node 0 wants to send a packet to node 3 with virtual coordinates $(c{+}k{+}1, c{+}k{+}1, c{+}k{+}1)$. When using the Euclidean distance we have $\delta_e(3,2) < \delta_e(3,1) < \delta_e(3,0)$ for $k > 1$. Therefore, node 0 would choose node 2 as its next hop. While considering the Hamming distance, we have $\delta_h(3,1) < \delta_h(3,2) < \delta_h(3,0)$ for $k > 1$; thus, node 0 selects node 1 as its next forwarder. Based on the virtual coordinates, node 1 is the best choice for forwarding since it minimizes the number of hops to reach the destination. For numerical example we can consider $c = 0$ and $k = 2$; thus, we have $\delta_e(3,2) = 1.73 < \delta_e(3,1) = 2 < \delta_e(3,0) = 2.44$ and $\delta_h(3,1) = 2 < \delta_h(3,2) = 3 < \delta_h(3,0) = 4$.

15.3.2 *Tree-Based Coordinate System*

As illustrated so far in this section, although simple and easy to implement, a plain hop-based coordinate system presents some drawbacks, especially regarding delivery ratio. In the following tree-based coordinate systems are developed with associated routing protocols to overcome the issues of classical landmark-based coordinate systems.

15.3.2.1 Tree-Based Coordinate System Construction

A tree-based coordinate system is built into two phases. In the first phase, a global election mechanism chooses a node that acts as the tree root. This node, also called root, initiates the tree construction. Nodes are assigned a unique ID by being labeled either through a depth-first search method or through a breadth-first search method. Routing is performed following the tree. Therefore, routing on top of trees guarantees message delivery since by definition, there exists exactly one path between any pair of nodes. However, the bootstrap phase may be much more complex than in a landmark-based coordinate system according to the protocols. In the following section, the different ways of assigning labels to nodes are explained.

15.3.2.2 Routing on Top of Trees

Liu and Abu-Ghazaleh [22] propose a stateless and guaranteed delivery georouting on tree-based virtual coordinates. They propose a one-dimensional virtual coordinate system based on a depth-first search pre-order traversal of the graph. A tree-style topology is constructed with only connectivity information. Starting from a root node with value 0, which may be randomly chosen, nodes are labeled by sending depth-first search packet to one of their neighbors. The node that receives the packet is assigned a unique identity which is the identity of the packet's sender incremented by 1. If a node does not have any unlabeled neighbor, it sends an end of search packet to its parent. As long as the network is connected, all nodes will eventually receive a unique identifier. Each node m is also given an interval $I(m) = [m, p]$ starting from its label to the greater identity of its children before traversal returns back to its parent. Routing is based on these labels, and current node forwards a packet to the node holding the interval containing the destination. Figure 15.15[1] shows the resulting tree building used for routing. The resulting tree depicted in the figure is based on an optimized and balanced construction of the tree using a breadth-first search algorithm.

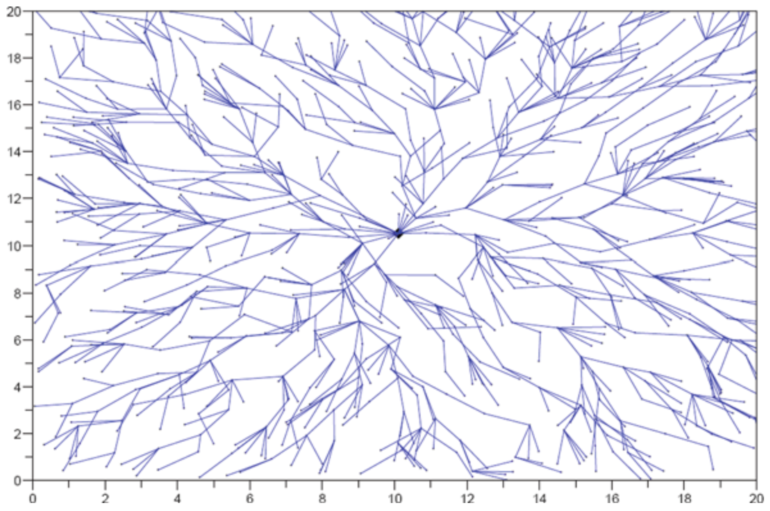


Fig. 15.15 Tree construction after the virtual coordinates assignment [22]

[1] Figure is taken from [22].

In the same way the proximal labeling process presented in [9] uses a depth-first traversal to build a tree. A flood tree is a tree obtained as follows. A root node is selected and it transmits a request message accepting up to a constant number, k, of replies. All the nodes accepted are linked to the root network, and the procedure is repeated recursively until all the nodes in the network are linked to the tree. After the flood tree is built the nodes are labeled in increasing order following a depth-first traversal of tree, leaving gaps between successive labels. Unlike [22] the skipping in the labeling procedure allows room for later insertions of new nodes.

The *LTP* protocol proposed in [8] also uses a tree for routing decision but in such a way that the path from any two nodes in the network is embedded in the label. In LTP, the tree is built iteratively from the root to the leaves. At bootstrap, a node is designed as root. This node may be a special node such as a fixed landmark or a selected node. At each step, every freshly labeled node queries its unlabeled neighbors and then gives a label to each answering node. If $l(u)$ is the label of node u and $|l(u)|$ the size of this label, the kth neighbor of node u is labeled $l(u)k$. Note that a node y already labeled as $l(y)$ may also answer to a node v to get a new label if and only if $|l(y)| > |l(v)| + 1$. The built tree gives the shortest path in the number of hops from the root to any other node. The distance used in the tree is based on label size and common prefix which can give the hop distance between any two nodes of the network. Thus the distance between node a and node b is $d_T(a, b) = ||l(a)| - |l(c)|| + ||l(c)| - |l(b)||$ where c is the lowest common ancestor of a and b and $|l(a)|$ (resp. $|l(b)|$) is the label size of node a (resp. of node b). Figure 15.16 shows an example of the labeling of LTP. However, this is worth noting that though reducing the stretch factor compared to other tree-based georouting protocols, LTP still presents a non-negligible stretch factor. This latter one can be even more reduced by using several trees and letting nodes following the most adequate tree at each routing decision. Indeed, additional trees mean additional bootstrap costs to build them. Nevertheless, studies have shown that using two trees

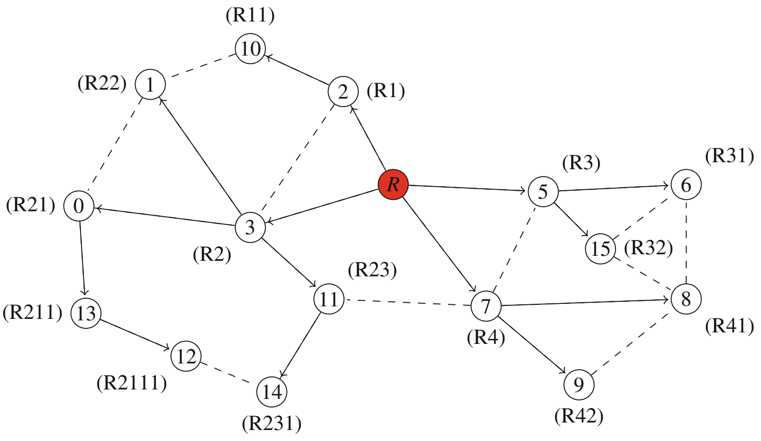


Fig. 15.16 Label results from the LTP protocol [8]

is enough to get a very low stretch factor. The main drawback of LTP (independently of drawbacks relative to tree construction and common to all protocols) is that if the tree is not well balanced, some labels may be very huge and generate a memory overhead.

The tree root is node 4 and has label R. Node 13 is labeled $R211$ since it is the first child of node 0 which has label $R21$. Node 3 has label $R2$ as the second child of the root. From Fig. 15.16 the distance between node 9 and node 5 is thus $d_T(9, 5) = ||l(9)| - |l(4)|| + ||l(4)| - |l(5)|| = |3 - 1| + |1 - 2| = 3$ hops.

As in every georouting protocol, each node is aware of the labels of itself, of its neighbors, and of the destination. Routing decision is then performed based on all these information and will try as much as possible to avoid to go through the root. The packet is forwarded to the neighbor whose d_T distance to destination is the lowest one. In this way, the routing path may follow "shortcuts" in the tree, decreasing the stretch factor. An example of such a case is given in Fig. 15.16 while considering the routing from node 9 to node 5. Node 7 sends directly to node 5, avoiding the tree root. This allows to reduce the stretch factor.

Algorithms based on tree constructions guarantee packet delivery but generate a much overhead, moreover resulting paths have high stretch factor since routes follow the tree and usually pass through the tree root.

In [37] the stretch factor is even more reduced but at the price of a huge message overhead and construction latency. In this paper, the authors propose *ABVCap*, an axis-based virtual coordinate assignment which is very close to tree-based coordinate system with multiple trees. In ABVCap, a node u is assigned at least one 5-tuple virtual coordinate $(u_{\mathrm{lo}}, u_{\mathrm{la}}, u_{\mathrm{rp}}, u_{\mathrm{up}}, u_{\mathrm{down}})$. According to its relative geographical position and node density (mean node degree), every node will be assigned one or more 5-tuple virtual coordinates (to an infinity). It is worth noting here that every node in the network does not have the same number of 5-tuple coordinate unlike all other proposals. The first two coordinates $(u_{\mathrm{lo}}, u_{\mathrm{la}})$ are used as location information. These locations are longitude and latitude. The three last coordinates $(u_{\mathrm{rp}}, u_{\mathrm{up}}, u_{\mathrm{down}})$ are used for routing. The coordinate assignment is split into four phases. In the first phase, four anchors (X,Y,Z,Z') are selected. In the second phase axes are established: latitude parallel (X–Y) and meridians (Z,Z'). These anchors and axis are fixed for the whole network and the same for every node. Based on these axis, some virtual meridians and parallels are virtually drawn over nodes through flooding from every anchor. Based on this, nodes are assigned their lo, la, and rp coordinates according to their relative distance to nodes that lie on meridian and/or parallel lines through a complex method. In the last phase, up and down coordinates are assigned. Figure 15.17 shows the resulting axis assignment of ABVCap.[2]

The routing process is then performed as follows. The routing packet contains the longitude and the latitude of the destination and a direction bit which is set to 1 if $s_{\mathrm{lo}} < d_{\mathrm{lo}}$ where s is the source and d the destination. Each node knows its multiple

[2] The figure is taken from [37].

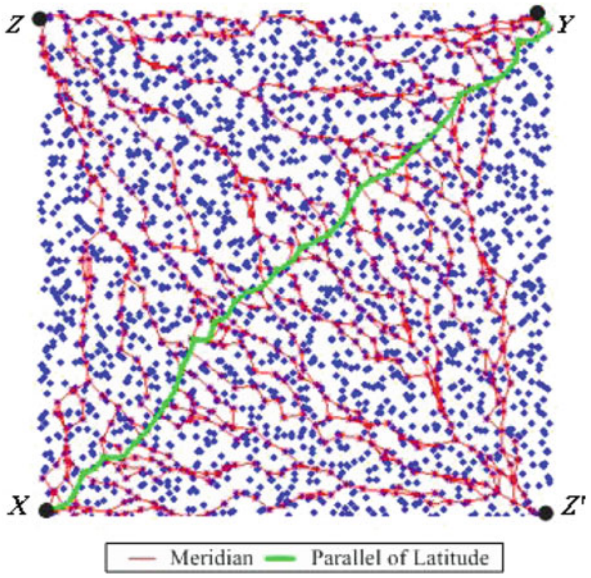


Fig. 15.17 ABVCap virtual coordinates assignment [37]

own coordinates, all multiple coordinates of all of its one-hop neighbors and of the destination. The routing decision is done in two phases. First, the source chooses the 5-tuple coordinates of the destination (among the set of coordinates of the destination) that minimizes the difference $s_{\text{lo}} - d_{\text{lo}}$. Second, the next hop choice is done based on Euclidean distance computed as $\delta_e(u, v) = \sqrt{(u_{\text{lo}} - v_{\text{lo}})^2 + (u_{\text{la}} - v_{\text{la}})^2}$. The authors prove that their routing protocol guarantees delivery and only use greedy forwarding such as in LTP. However, compared to LTP, the stretch factor[3] of ABVCap is lower and it is more robust to changes in the network, but the initialization phase is costly in number of messages and latency.

So far in this section, literature has witnessed enhancements in terms of guaranteed delivery to the detriment of stretch factor (and thus energy consumption and bandwidth overhead). In addition, energy efficiency has been poorly addressed.

15.3.2.3 Energy Efficiency

In this section, we provide an in-depth description of *HECTOR* [24] which is, to the best of our knowledge, the only georouting protocol that both is energy efficient and guarantees packet delivery on top of virtual coordinates. This protocol mixes the use of tree-based coordinate system and landmark-based coordinate system and therefore highlights the previous sections.

[3] The stretch factor is the difference in route length between the one computed by the algorithm and the optimal path.

The routing protocol and virtual coordinate assignment used in [24] take benefits from both tree-based labels like in [8] and landmark-based coordinates like in VCap [7] or VCost [11], with the use of Hamming distance to provide a routing solution based on virtual coordinates with a short initialization phase. The aim of this work was to provide a georouting algorithm with no position information which guarantees delivery and is energy efficient. The authors of [24] propose two algorithms according to the ability of nodes to adjust their range: *HECTOR'* when nodes cannot tune their transmission range and HECTOR when they can. *HECTOR* is thus also energy efficient. Both algorithms use both kinds of coordinates: LTP labels and landmark-based labels.

The routing decision of HECTOR' (resp. HECTOR) is done in such a way that the tree label distance (computes on T labels, noted d_T like for LTP) is always decreasing or at least stationary, and the V coordinates (VCap-like coordinates, Hamming distance on V coordinates is noted d_V) are used to reduce the stretch factor. The basic idea is the following. A source node s holding a packet for a destination node d performs a greedy routing scheme in a VCap (resp. VCost) fashion for HECTOR' (resp. HECTOR). In order to avoid to be trapped in a local minimum, the routing algorithm selects the next hop with regard to not only the V coordinates but also the T labels. The routing process runs as follows. When node u receives a message for node d, it first considers its neighbors in the forward direction, based on both their T and V coordinates. It only considers node v for which d_T distance (distance based on the tree) toward d is equal or smaller than the tree distance between u and d ($d_T(v, d) \leq d_T(u, d)$). We note $N_T(u)$ the set of neighbors of node u such that $d_T(v, d) \leq d_T(u, d)$. Such neighbors always exist ($N_T(u) \neq \emptyset$) if the network is connected because of the tree construction. The node u first checks whether any one of these nodes also provides a progress with respect to V coordinates. We note $N_V(u)$ the set of neighbors of u providing a progress toward the destination with regard to V coordinates, i.e., such that $d_V(v, d) \leq d_V(u, d)$. If $H = N_T(u) \cap \{N_V(u) \cup v \, | d_T(v, d) = d_T(u, d)\}$ is the set of such nodes and $H \neq \emptyset$ then u selects its next hop among the nodes in H thus reducing the distance (resp. optimizing the cost over progress) toward the destination regarding coordinates V and not increasing distance regarding T labels. If $H = \emptyset$, node u selects its neighbor v which provides the best progress to the destination regarding T labels. Such a node always exists since there always exists exactly one path in the tree between any two nodes. In case of ties, the next hop is chosen at random between candidates.

As already mentioned, the progress in HECTOR is computed based on a cost-over-progress fashion. If $H \neq \emptyset$ then u selects its next hop among the nodes in H as the node v which provides the best ratio cost over progress to the destination regarding the virtual coordinates (v such that $\text{COP}_V(u, v, d) = \min_{w \in N_V(u)} \text{COP}_V(w)$). If $H = \emptyset$, v is chosen such that $\text{COP}_T(u, v, d) = \min_{w \in N_T(u)} \text{COP}_T(u, w, d)$. This variant needs the distance between nodes to evaluate the cost of the transmission.

Let us take Figs. 15.11 and 15.16 to illustrate the behavior of HECTOR and HECTOR'. The routing between node 14 (L_3) and node 5 gives an example of the guaranteed delivery provided by HECTOR' and HECTOR. Following VCap or VCost scheme, node 11 is the next hop chosen by node 14. In VCap or VCost, node

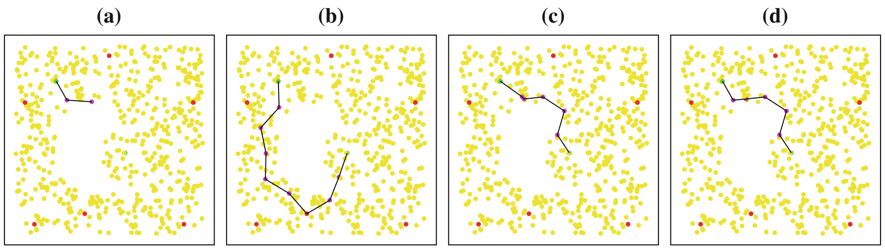


Fig. 15.18 Illustration of the paths followed by each algorithm with the use of five landmarks. VCost/VCap fails after the second hop, and LTP passes through the tree root. HECTOR and HECTOR' combine both T and V coordinates: (**a**) VCost, (**b**) LTP, (**c**) HECTOR', (**d**) HECTOR

11 is a dead end if we use Hamming distance since distance from node 11 to node 5 is equal to 2 and distance from node 7 to node 5 is also 2. As HECTOR' and HECTOR use T labels to avoid reaching a dead end, node 3 is chosen as the next hop of node 11 since $d_T(3,5) < d_T(11,5)$. Packet will then go through node 4 as its last relay to node 5.

Figure 15.18 shows the path shapes of different routing algorithms depending on the coordinate system used. This figure shows how HECTOR' and HECTOR take benefit from T labels and V coordinates to provide a guaranteed delivery routing with an enhanced stretch factor.

The following theorems and lemmas prove that (in Lemma 1) at each step of the algorithm there is always a progress at least on T coordinates, that the resulting path is loop free (Lemma 2), and that there is always a next hop that is closer to the destination than the current node (Lemma 3). These lemmas lead to Theorem 3 which shows that the routing process guarantees packet delivery.

Lemma 1 *A packet cannot transit from a node u to another node v if $V(u) = V(v)$ (or if $d_V(u,d) < d_V(v,d)$) unless there is a positive progress regarding T coordinates (if $d_T(u,d) > d_T(d,v)$).*

Proof Let us assume that node u holds a packet for a destination w. Suppose that nodes u and v have the same V coordinates ($V(u) = V(v)$) or that v is farther than node u regarding the V coordinates ($d_V(u,w) < d_V(v,w)$), then by definition node $v \notin N_V(u,w)$. Thus $v \notin H$. The selected next hop is thus part of $H' = \{x|\text{COP}_T(u,x,w) = \min_{i \in N_T(u)} COP_T(u,i,w)\}$, which only contains neighbors of u closer to w than u regarding T labels. Thus, if node v is chosen as the next hop, that means that $v \in H'$ and thus provides a progress regarding T coordinates. Note that in the worst case, i.e., when the progress on T coordinates is minimal, the next hop is either the parent or a child of node u. ■

Lemma 2 *HECTOR is loop free.*

Proof Let us assume that node u_0 is the source of a packet, w its destination, and node u_1 the next hop chosen by node u_0. This means that $d_T(u_0,w) \geq d_T(u_1,w)$ is based on H and H' construction. It suffices to show that the next hop chosen by node u_1 cannot be node u_0.

- CASE (1) Let us first assume that node $u_1 \in N_V(u_0, w)$. This means that $d_V(u_1, w) < d_V(u_0, w)$, thus $u_0 \notin N_V(u_1, w)$. Thus, node u_0 could be selected as the next hop of u_1 if and only if it provides a progress regarding T coordinates $d_T(u_0, w) < d_T(u_1, w)$ (Lemma 1). But according to Lemma 1, if u_1 is the next hop chosen by u_0, we have $d_T(u_1, w) \leq d_T(u_0, w)$. We thus reach a contradiction and thus, the next hop chosen by node u_1 cannot be node u_0.
- CASE (2) Let us now assume that node $u_1 \notin N_V(u_0, w)$, Lemma 1 tells us that u_1 is a parent, a child, or a node that maximizes the cost over progress toward w from node u_0 thus, if u_1 is the next hop for u_0, $d_T(u_0, w) > d_T(u_1, w)$ and node u_0 cannot be selected as the next hop for node u_1.

By transitivity of $d_T()$ and $d_V()$ we cannot have a path $u_0, ..., u_i, ..., u_O$. These two cases imply that the routing protocol is loop free. ∎

Lemma 3 *Using HECTOR's coordinate system, there always is a next hop that is closer to the destination regarding virtual coordinates.*

Proof Let us consider a source u and a destination w. By construction, if a node in $N_V(u, w)$ is chosen as the next hop, this ensures a progress in the V coordinates. If the next hop is chosen in $N_T(u, w)$ this ensures a progress in the tree toward the destination. ∎

It is worth noting that the progress made on V coordinates is more important than the progress made on T coordinates in the geographical space. Indeed, the next hop in the T labels can have the same V coordinates and thus more or less the same Euclidean distance to the destination. The greedy aspect provided by this algorithm makes it simple, memoryless, and scalable. It is interesting to see here how the simple combination of two coordinates system can enhance the performances (path length, energy efficiency, guaranteed delivery) of georouting protocols. It is worth noting that the authors of [24] mainly focus their work on energy efficiency by the use of cost over progress for next hop selection. As a result, the idea defended by authors can be applied to reduce the hop distance, the Euclidean distance, and any metric for next hop selection. The authors also highlight that V coordinates can be replaced by real geographic routing if real geographic coordinates are available.

Theorem 3 *HECTOR guarantees packet delivery.*

Proof Each node has a unique label due to the labeling process. This ensures that the destination of a packet is unique and that at each step of the routing protocol, a next hop closer to the destination can be found. Based on Lemma 1, Lemma 2, and Lemma 3 if a path exists (if the network is connected), the routing protocol will find one in a greedy way. ∎

15.3.2.4 Tree-Based Positioning System Issues

The main drawbacks of tree-based coordinate systems are the building process and tree maintenance. First, the root's choice can strongly impact the tree shape and thus the routing performance in terms of path length. Second, since tree construction is associated with a labeling process and is done in a depth-first search of a breadth-first search process, the label's size may have different sizes for each node. Finally, the tree maintenance is a difficult task since the tree construction is based and initialized by centralized nodes; if a node dies, the whole tree has to be rebuilt.

15.4 Conclusion

Indeed, in this chapter, we have reviewed most of the literature georouting protocols. Table 15.1 sums up the different categories and algorithms, with respect to their characteristics.

Table 15.1 Classification of georouting protocols

	Exact position	Virtual position
Hop based	Greedy [14], MFR [35]	VCap [7], Gliders [13]
Directional	DIR [18], DREAM [1], LAR [17]	
Energy efficient (EE)	COP [19], NFP [16]	VCost [11]
Guaranteed delivery (GD)	GFG [4]	LTP [8], [22], ABVCap [37]
EE+GD	SPFSP [31], EtE [12]	HECTOR [24]

Energy-efficient algorithms assume that nodes are able to compute the Euclidean distance between themselves and their neighbors. Indeed, each algorithm presents its strengths and weaknesses.

Most of the time, there is no more suitable algorithm. Trade-offs have to be made with respect to the context, their environment, and node abilities: geographical or virtual coordinates, abilities to compute an Euclidean distance, computing resources, memory size, etc.

Wireless links are prone to multiple physical phenomena such as interference, collisions, shadowing, fading. They also are impacted by obstacles, buildings, and meteorology conditions. All these features make the transmission unpredictable and unreliable. Thus, next steps of research will be to cope the different algorithm characteristics mentioned in this chapter with the impact of their application in real wireless environments.

References

1. S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward. A distance routing effect algorithm for mobility (dream). In: *Proceedings of the* 4*th ACM Annual International. Conference on Mobile Computing and Networking (MOBICOM)*, pages 76–84, Dallas, Texas, 1998.
2. F. Benbadis, J-J. Puig, M.Dias de Amorim, C. Chaudet, T. Friedman, and D. Simplot-Ryl. Jumps: Enhanced hop-count positioning in sensor networks using multiple coordinates. *International Journal on Ad Hoc & Sensor Wireless Networks*, 2008.
3. P. Boone, E. Chavez, L. Gleitzky, E. Kranakis, J. Opartny, G. Salazar and J. Urrutia Morelia Test. Improving the efficiency of the gabriel test and face routing in Ad-hoc Networks *Lecture Notes in Computer Science* , 3104:23–24, 2004, 2008.

4. P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. In: *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communication (DIAL-M)*, pages 48–55, Seattle, WA, August, 1999.
5. J. Carle and D. Simplot-Ryl. Energy efficient area monitoring by sensor networks. *IEEE Computer Magazine*, 37:40–46, 2004.
6. S. Capkun, M. Hamdi and J-P Hubaux. GPS-Free positioning in mobile ad-hoc networks In: *Proceedings of the Hawaii International Conference on System Sciences HICSS*, Hawaii, USA, 2001, 2005.
7. A. Caruso, S. Chessa, S. De, and A. Urpi. Gps free coordinate assignment and routing in wireless sensor networks. In: *Proceedings of the 24th Conference of the IEEE Communications Society (INFOCOM)*, volume 1, pages 150–160, Miami, USA, March, 2005.
8. E. Chávez, N. Mitton, and H. Tejeda. Routing in wireless networks with position trees. In: *Proceedings of the 6th International Conference on AD-HOC Networks & Wireless (Ad Hoc Now)*, Morelia, Mexico, September 2007.
9. E. Chávez, M. Fraser, and H. Tejeda. Proximal labeling for oblivious routing in wireless ad hoc networks. In *ADHOC-NOW '09: Proceedings of the 8th International Conference on Ad-Hoc, Mobile and Wireless Networks*, Springer, Berlin, pages 360–365, 2009.
10. T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum, and L. Viennot. Optimized link state routing protocol (OLSR), 2003. RFC 3626.
11. E. H. Elhafsi, N. Mitton, and D. Simplot-Ryl. Cost over progress based energy efficient routing over virtual coordinates in wireless sensor networks. In: *Proceedings of IEEE International Workshop: From Theory to Practice in Wireless Sensor Networks (t2pWSN)*, Helsinki, Finland, 2007.
12. E.H. Elhafsi, N. Mitton, and D. Simplot-Ryl. End-to-end energy efficient geographic path discovery with guaranteed delivery in ad hoc and sensor networks. In: *Proceedings of the 19th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Cannes, France, September 2008.
13. Q. Fang, J. Gao, L.J. Guibas, V. de Silva, and Li Zhang. GLIDER: gradient landmark-based distributed routing for sensor networks. In: *Proceedings of the 24th Conference of the IEEE Communications Society (INFOCOM)*, volume 1, pages 339–350, Miami, USA, March 2005.
14. G.G. Finn. Routing and addressing problems in large metropolitan-scale. *Internetworks, ISI Research Report ISU/RR-87-180*, March 1987.
15. H. Frey, and I. Stojmenovic. On delivery guarantees of face and combined greedy-face routing in ad hoc and sensor networks. *Proceedings of the 12th ACM Annual International Conference on Mobile Computing and Networking (MOBICOM)*, Los Angeles, CA, ages 390–401, September 2006.
16. T.-C. Hou and V. Li. Transmission range control in multihop packet radio networks. *Communications, IEEE Transactions on [legacy, pre - 1988]*, 34(1):38–44, 1986.
17. Y-B Ko and N. H. Vaidya. Location-aided routing (lar) in mobile ad hoc networks. In: *Proceedings of the 4th ACM Annual Int. Conference on Mobile Computing and Networking (MOBICOM)*, pages 66–75, Dallas, Texas, 1998.
18. E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. In: *Proceedings of the 11th Canadian Conference on Computational Geometry (CCCG)*, Vancouver, Canda, 1999.
19. J. Kuruvila, A. Nayak, and I. Stojmenovic. Progress and location based localized power aware routing for ah hoc sensor wireless networks. *IJDSN*, 2:147–159, 2006.
20. J. Li, L. Gewali, H. Selvaraj and V. Muthukumar. Hybrid Greedy/Face Routing for Ad-Hoc Sensor Networks. In *DSD Journal*, Los Alamitos, CA, 0:574–578, 2004.
21. X. Lin and I. Stojmenovic. Geographic distance routing inad hoc wireless networks. Technical Report TR-98-10, SITE, University of Ottawa, December 1998.
22. K. Liu and Nael Abu-Ghazaleh. Stateless and guaranteed geometric routing on virtual coordinate systems. In: *Proceedings of the IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, pages 340–346, October 2006.

23. Ke Liu and Nael Abu-Ghazaleh. Aligned virtual coordinates for greedy routing in wsns. In *Proceedings of the IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, pages 377–386, October 2006.
24. N. Mitton, T. Razafindralambo, D. Simplot-Ryl, and I. Stojmenovic. Hector is an energy efficient tree-based optimized routing protocol for wireless networks. In *Proceedings of the 4th International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, Wuhan, China, December 2008.
25. N. Mitton, D. Simplot-Ryl, and I. Stojmenovic. Guaranteed delivery for geographical anycasting in wireless multi-sink sensor and sensor-actor networks. In: *Proceedings of the 28th Conference on Computer Communications (INFOCOM)*, Rio de Janeiro, Brasil, April 2009.
26. Nelson, R. and Kleinrock L. The spatial capacity of a slotted ALOHA multihop packet radio network with capture In *IEEE Transactions on Communications*, 32, 6: 684–689, June 1984.
27. D. Niculescu and B. Nath. Ad hoc positioning system (APS). In: *Proceedings of IEEE Global communications conference GLOBECOM*, San Antonio, USA, 2001.
28. N. Patwari, J.N. Ash, S. Kyperountas, A.O. III Hero, R.L. Moses, and N.S. Correal. Locating the nodes: Cooperative localization in wireless sensor networks. *Signal Processing Magazine, IEEE*, 22(4):54–69, July 2005.
29. C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector routing, 2003. RFC 3561.
30. A. Rao, C. H. Papadimitriou, S. Shenker and I. Stoica. Geographic routing without location information, *Proceedings of the 9th ACM Annual International Conference on Mobile Computing and Networking* (*MOBICOM*), 96–108, 2003.
31. J.A. Sanchez and P.M. Ruiz. Exploiting local knowledge to enhance energy-efficient geographic routing. In: *Proceedings of the 2nd International Conference on Mobile Ad-hoc and Sensor Networks MSN*, pages 567–578, December 2006.
32. I. Stojmenovic, S. Datta. Power and cost Aware Localized Routing with guaranteed delivery in wireless networks. *Wireless Communication and Mobile Computing*, 2(4):175–188, 2004.
33. I. Stojmenovic and X. Lin. Loop-Free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks *IEEE Transactions Parallel Distribution Systems (TPDS)*, 12(10):1023–1032, 2001.
34. I. Stojmenovic and X. Lin. Power-aware localized routing in wireless networks. *IEEE Transactions Parallel Distribution Systems (TPDS)*, 12(11):1122–1133, 2001.
35. H. Takagi and L. Kleinrock. Optimal transmission ranges for randomly distributed packet radio terminals. *IEEE transaction on communications*, com-22(3):246–257, 1984.
36. G. Toussaint. The relative neighborhood graph of a finite planar set. *Pattern Recognition Journal*, 12(4):261–268, 1980.
37. M-J Tsai, H-Y Yang, and W-Q Huang. Axis-based virtual coordinate assignment protocol and delivery-guaranteed routing protocol in wireless sensor networks. In: *Proceedings of the 26rd Conference of the IEEE Communications Society (INFOCOM)*, volume 1, pages 2234–2242, May 2007.
38. G. Voronoi. New applications, from continuous parameter to quadratic shape theory. *Journal für die Reine und Angewandte Mathematik*, 133:97–178, 1907.
39. Y. Wang, W.-Z. Song, W. Wang, X.-Y. Li, T.A. Dahlberg. LEARN: Localized energy aware restricted neighborhood routing for Ad Hoc networks. In: *Proceedings of the 3rd Annual IEEE Sensor and Ad Hoc Communications and Networks(SECON)*, 2, 508–517, 2006.
40. T. Watteyne, I. Augé-Blum, M. Dohler, S. Ubeda, and D. Barthel, Centroid virtual coordinates - A novel near-shortest path routing paradigm. To appera in *International Journal of Computer and Telecommunications Networking, Elsevier*, 53(10):1697–1711, to appear.