Chapter 4. Geographic Routing in Wireless Sensor and Actuator Networks

Abstract
Position information enables development of localized routing methods where greedy routing decisions are made at each node, based solely on knowledge of positions of neighbors and destination, with considerable savings in communication overhead. Power consumption can be taken into account in the routing process. This chapter will survey existing flooding based and position based routing schemes. It also describes a general cost to progress ratio based approach for designing routing protocols under a variety of metrics, such as hop count, power, remaining energy, delay, and others. Chapter describes also routing with guaranteed delivery for unit disk graphs and ideal MAC layer. Gabriel graph, as localized planar and connected structure needed for such solutions is described. Solutions are expanded toward beaconless behavior, where nodes are not aware of their neighborhood. Georouting with virtual coordinates is based on hop distances to some landmarks. This chapter also discusses physical layer aspects of georouting, routing in sensor-actuator networks, and load balancing issue in routing.

4.1. Flooding based routing and georouting in sensor networks
Sensors report their measurement to a monitoring station, called also base station, or simply sink. Individual reporting of discovered events are normally done by a routing task, from a sensor to the sink. In a routing task, a packet is to be sent from a source node to a destination node, via some intermediate nodes in a given multi-hop network. In wireless sensor networks, source is normally a sensor while the destination is a sink. In sensor-actuator networks, actuator (actor) may serve as source and/or destination node. Sinks, actuators and even sensors could be fixed or mobile.

Existing practical implementations of sensor networks normally avoid the use of position information due to current technological difficulties in providing it to sensors with sufficient accuracy. Routing is then based on flooding as a step to find route. Flooding was covered in Chapter 2. It will be reviewed again in Chapter 6 to illustrate data gathering and data aggregation by constructing a tree centered at the sink/actuator. Monitoring center floods route discovery (short) message to all sensors located inside a region. Sensors establish links toward sensor from which the first copy of packet is received, and use that link for reporting, or forwarding reports received from neighbors. Thus effectively sensors report along reverse broadcast tree. In wireless ad hoc networks, this method is currently a candidate for being a standard, with mobile nodes such as laptops, palmtops, mobile phones, etc. serving as sources, destinations, and intermediate nodes. Source then floods the network, and destination node replies back to source upon receiving discovery message(s) using memorized hops (AODV) [PRD99] or paths (DSR) [JM96]. This method was applied to sensor networks, and is often cited as ‘directed diffusion’ following description under such a name in [IGE00]. There are many variants of this basic method, discussing also multi-path construction, quality of service provision etc. Route discovery message may contain accumulated delay, congestion, power (that is, a pre-specified cost metric) along paths. The best path is then selected at the destination node. A number of local route maintenance techniques are proposed to handle dynamic nature of the network; however the maintenance is generally expensive, and often is done by triggering another network wide flooding. To reduce the area being flooded, the
expanding ring search is often applied, which floods in restricted area hoping that the destination will be found there; otherwise the searched area size is increased (e.g. doubled).

Ding, Sivalingam, Kashyapa, and Chuan [DSKC03] considered the problem of finding a route from a sensor to the single sink in a wireless sensor network. Following a reactive route discovery strategy, the sink floods the network and sets the routes. The difference is that each sensor does not memorize the whole route, or a single pointer to predecessor sensor on the route, but instead it memorizes its hop count distance to the sink. When a packet is sent toward the sink, any neighbor at one less hop distance can forward it, instead of reporting back to the first node that sent task assignment packet to it. For instance, report can be sent to the neighbor with highest energy and smaller hop count, or any neighbor that sent packet with smaller hop count from the sink [DSKC03]. Node can memorize few such alternatives during setup phase and try them one by one. Alternatively, a neighbor at one less hop distance can simply retransmit, and node can block further retransmissions by a separate blocking packet.

Since the geographical location of an event is an important information, position information of sensors is considered available, while recognizing difficulty in gathering it with reasonable accuracy. Geographic routing (georouting) is the strategy which employs geography information of nodes when routing from the source to the destination. It assumes that nodes in the networks are provided with GPS (Global Positioning System) devices [HLC97] or some localization techniques [BT05] are available to obtain location information of nodes. The nodes exchange location information with their neighbors, and forward packets based on location information of their neighbors and destination. It allows routers to be nearly stateless, since packet forwarding is based on location information of candidate neighbors and the location of the final destination only. In WSANs (wireless sensor-actuator networks), the destination is normally a sink or an actuator. Location of the destination is flooded over the network to all sensors at initialization stage of the system. Since nodes are not required to maintain routing tables and routing decisions are made based on geographic information, the routing information grows as the density of the network, e.g. average node degree, rather than size of the network, e.g. total number of nodes. Therefore, the geographic routing algorithms are normally characteristic of low computation complexity and high scalability which are desirable in large-scale wireless networks. By their localized nature, geographic routing algorithms are highly scalable solutions that do not require any additional control overhead when network topology changes due to energy-conserving sleep cycles [FS05]. However, highly mobile networks are difficult to handle since geographic routing is normally based on locations of the destination and neighboring nodes. For instance, obtaining an accurate location of a mobile destination is even more difficult than routing itself [S02]. This chapter will deal only with the case of static sinks as or actuators destinations.

4.2 Greedy, Projection and Direction Based Routing

The simplest form of geographic routing is Greedy routing which was first described by Finn in 1987 [F87]. In the greedy routing algorithm, each node in the route forwards packets to the neighbor which is the closest to the destination among its neighbors. Only the neighbors that are closer to the destination than the current node are considered. The
algorithm is illustrated in Fig. 4.1. Suppose node $S$ is the current node in the route and node $D$ is the destination. All neighbors of $S$ are within the circle centered at $S$. Suppose node $B$ is the closest to destination $D$ among all $S$’s neighbors and $d$ is the distance between $B$ and $D$. According to the greedy routing algorithm, node $S$ selects node $B$ as the next hop forwarding node. Greedy routing is suitable for large-scale networks with high density and frequent topology changes because it is simple and localized. The distance to the destination is minimized in each hop of the routing. The $GEDIR$ [SL01a] algorithm is a variant of Greedy routing. It considers all neighbors (even in backward direction) and selects the node that is the closest to the destination. A message is dropped if it would be sent back to the node where it was previously received from. Another variant is to select nearest neighbor among those that are closer to destination than current node (NC method, proposed in [SL01b]).

![Fig. 4.1 Greedy, projection and direction based routing.](image)

The first geographic routing was described by Takagi and Kleinrock [TK84]. The notion of progress was introduced to define the most forward within radius (MFR) greedy routing algorithm. Suppose $A$ is a neighbor of $S$. The progress of $A$ to $D$ corresponds to the dot product $SA \cdot SD = |SA||SD|$ of two vectors $SD$ and $SA$, where $SA'$ is the projection of $SA$ onto line $SD$, and $|XY|$ is the Euclidean distance between $X$ and $Y$ (dot product formulation of the progress is given in [SL01a]). The MFR algorithm considers all neighbors of $S$ and selects the node $A$, such that $SA \cdot SD$ is maximized. That is, the (signed) length of projection $|SA'|$ is maximal. Then, node $S$ forwards packets to node $A$. Only neighbors with positive progress are considered in MFR. The nearest neighbor with forward progress (NFP) method [HL86] selects nearest neighbor among nodes with positive progress.

Another strategy of geographic routing utilizes direction information of next hop candidates with respect to line toward the destination. Kranakis, Singh and Urrutia proposed compass routing (also referred to as the DIR method) in [KSU99]. It selects the next hop by minimizing the angle $\angle ASD$ between lines toward candidate neighbor $A$ and destination $D$. In the example in Fig. 4.1, $\angle CSD$ is the minimum such angle among all $S$’s neighbors, and node $C$ is selected as the forwarding node.

Stojmenovic and Lin [SL01a] proved that Greedy, GEDIR and MFR routing are loop-free while DIR routing is not. Greedy routing selects the neighbor which is closer to the destination than the current node. There is no backtracking and thus it is loop-free. The proof that MFR is loop free follows. Suppose $A_1, A_2, \ldots, A_n$ are the nodes in the loop so
that $A_1$ forwards packets to $A_2$, $A_2$ forwards packets to $A_3$, …, $A_{n-1}$ forwards packets to $A_n$ and $A_n$ forwards packets to $A_1$. At current node $S$, MFR selects neighbor $A$ for which $AD\cdot SD$ is minimized, i.e. $SA \cdot SD$ is maximized. We have $DA_n \cdot DA_1 < DA_2 \cdot DA_1 = DA_1 \cdot DA_2$ since node $A_1$ forwards packets to $A_2$ not $A_n$. Similarly, it follows $DA_1 \cdot DA_2 < DA_2 \cdot DA_3 < \ldots < DA_{n-1} \cdot DA_n < DA_n \cdot DA_1$. This is a contradiction. Therefore, MFR routing is loop-free.

Transmission radius

Fig. 4.2 A counter example for direction based routing.

The example in Fig. 4.2 shows that the direction based routing can not guarantee loop-free routes in UDGs. Suppose $S$ is the source node, and $B$ is not its neighbors (transmission radius is shown in the figure). According to the direction based routing, node $S$ selects neighbor $A$ since its deviation from the line $SD$ is smaller than that of $C$. Similarly, node $A$ selects $B$, $B$ selects $C$, and $C$ selects $S$. Thus, the direction based routing enters the loop $S \rightarrow A \rightarrow B \rightarrow C \rightarrow S$.

Five routing algorithms: greedy, MFR, direction (compass) routing, the shortest path routing (in terms of hop count) and the NC (nearest closer) routing are illustrated in Fig. 4.3. The task is to find a route from the source $S$ to the destination $D$.

Fig. 4.3 Paths with different routing algorithms: Greedy: $S \rightarrow C \rightarrow U \rightarrow F \rightarrow I \rightarrow N \rightarrow D$, MFR: $S \rightarrow C \rightarrow U \rightarrow F \rightarrow I \rightarrow M \rightarrow D$, DIR: $S \rightarrow T \rightarrow E \rightarrow G \rightarrow I \rightarrow N \rightarrow D$, The shortest path: $S \rightarrow T \rightarrow G \rightarrow I \rightarrow N \rightarrow D$, NC: $S \rightarrow B \rightarrow C \rightarrow U \rightarrow F \rightarrow H \rightarrow I \rightarrow N \rightarrow D$.

4.3. Applications of Cost to Progress Ratio Framework to Georouting
Stojmenovic [S06] proposed a framework for designing network layer protocols for sensor networks including localized routing, broadcasting, area coverage, and so on. The framework is based on optimizing the ratio of the cost to progress, where the cost to reach the next hop forwarding node in routing is expressed in a certain metric, and the progress is a measure of advance towards the destination.

Examples of cost metric are hop count, power, reluctance, power*reluctance, delay, and expected hop count [S06] (see also Chapter 1). Each link has a cost measure which depends on the assumptions and metrics used. The framework assumes that each node knows the cost of each of its links to neighboring nodes. The basic idea of the framework is as follows. Suppose the source or current node $S$ has $k$ neighbors, where only neighbors closer to the destination than the current node are considered to ensure progress at each step. That is, $S$ has $k$ choices to forward a packet towards the destination. Node $S$ then computes $C_i/P_i$, $i=1, 2, \ldots, k$ for each neighbor, where $C_i$ and $P_i$ are the cost and progress, respectively, of $i$-th candidate neighbor. The neighbor with the minimum cost to progress ratio is selected to forward the packet. The same rule is continuously applied by receiving node to select the next hop. The routing process continues until the destination is reached or no neighbors with progress are available. If no such neighbor exists, the packet is dropped or a recovery scheme, based on the specific cost metric used, is applied to make a progress before resuming the scheme. The framework was illustrated by applying it to the following well-known geographic routing algorithms.

In the greedy routing [F87] introduced in the previous section, the cost metric is the hop count (the number of transmissions on a route) and the progress made by forwarding is reduction of distance to the destination. For the current node that holds a packet, the cost to transmit to any of its neighbors is the same, one hop. In the example in Fig. 4.1, the cost to progress ratio for node $B$ is $1/(|SD| - |BD|)$. $|SD| - |BD|$ is to be then maximized, as in greedy routing algorithm. If the progress metric is defined as the projection of neighbors on line $SD$, the routing algorithm becomes the MFR [TK84].

Another example is the localized power-aware routing which was described by Kuruvila, Nayak and Stojmenovic [KNS04]. In Fig. 4.4, the power required for node $C$ to reach node $A$ is proportional to $|CA|^\alpha+c$, where $\alpha$ is power attenuation factor which is normally between 2 and 6 and $c$ is a constant. Constant $c$ accounts for the energy and minimal signal strength for correct signal reception. This power measure is used as the cost. The progress is defined as $|CD|-|AD|$ (only positive progress is considered, $|CD|>|AD|$). Thus, the cost to progress ratio of the power-aware routing is $(|CA|^\alpha+c)/(|CD|-|AD|)$. The selected neighbor minimizes the power spent per unit of progress made in terms of getting closer to the destination.
Note that power-aware routing may result in early energy depletion of certain nodes. If residual energy of nodes is included into cost as reluctance, the goal of routing is to maximize the number of routing tasks the network can perform. For instance, let \( f(A) \) denote the inverse \( 1/g(A) \) of the normalized (in interval [0,1]) residual energy \( g(A) \) in node \( A \). Nodes with more residual energy, i.e. smaller \( f(A) \), are more eager to forward packets while nodes with less residual energy, i.e. larger \( f(A) \), are reluctant to do so. The routing algorithm selects neighbor \( A \) that minimizes \( f(A)/(|CD|−|AD|) \), subject to \(|CD|>|AD|\). This routing framework can be applied to other cost metrics, such as QoS requirements, transmission delay, link quality and data.

All routing protocols based on the cost to progress ratio can be improved by applying the iterative improvement method which was described by Huang, Dai and Wu [HDW04] (for QoS metric costs) and [KNS04]. Suppose current node \( C \) selects neighbor \( A \) to minimize the cost to progress ratio while the overall goal is to minimize the total sum of costs over the route. If there is another neighbor \( B \) of node \( C \), such that \( \text{cost}(CB) + \text{cost}(BA) < \text{cost}(CA) \) then it could be more beneficial to forward the packet to node \( B \) instead of node \( A \). Such improvement could be iteratively repeated until no improvement is possible. Note that the improvement may be locally done at node \( C \) without message exchange if node \( C \) has needed information for given metric (it involves metric between two neighbors).

The iterative improvement is a special case of the general scheme [SR06, WC07, EMS08] (with power consumption as the metric) based on shortest weighted paths toward temporary destination. Suppose that current node \( C \) selects neighbor \( A \) having the best cost to progress ratio (in the first proposal, [SR06], temporary destination is decided by hop count based greedy routing). Instead of sending the message directly to \( A \), node \( C \) constructs the shortest cost path (using the same cost metric as weight) from \( C \) to \( A \), and forwards the packet to the first node \( B \) on that path (often \( B=A \)) to minimize the overall cost. That node then applies the same reasoning, starting from selecting its own temporary destination [EMS08], or keeping the same destination until it is reached [WC07]. To avoid loops between two neighboring nodes, only neighbors directly connected to temporary destination are considered. If temporary destination is fixed until it is reached [WC07] then progress toward final destination at each step is not verified. Otherwise, to avoid loops, only nodes closer to final destination (not only temporary one) could participate in the shortest path [EMS07].

If greedy routing (with a given cost metric) cannot make progress at a given node, recovery mode is invoked. Recovery mode (covered later in this chapter) uses hop count as metric to guarantee recovery (such resolution is proposed in [SD04]). However, the total cost of following these predetermined edges is then not optimized with respect to given cost metric. In algorithms [WC07, EMS08], these edges are indirectly followed, by replacing direct transmission between endpoints of these edges with shortest weighted paths between them. Further, [EMS08] builds a connected dominating set from given set of nodes, and computes its Gabriel graph to obtain the planar graph \( G' \). Face routing is applied on \( G' \) only to decide which edges to follow in the recovery process. On each edge, shortest weighted path routing is applied. Then the next edge is similarly followed, until recovery is possible. This two-phase (greedy-Face) routing process reiterates until the final destination is reached.

One of major advantages of cost-to-progress ratio framework is that it has no added
parameters such as thresholds. In typical threshold based approach, ‘bad’ links are eliminated, and packet is dropped if there is no ‘good’ neighbor. However, a reasonable path may contain only one weak ‘bridge’. Experiments so far indicate that threshold based approaches are inferior for all threshold values, because of either high failure rate if threshold is too restrictive, or suboptimal path choices when generous threshold choice allows one or more very weak links into a path, creating a bottleneck for the route. This occurs because final selection over ‘acceptable’ links is made normally by using a metric different from the cost metric. For example, node closest to the destination could be chosen, despite its barely acceptable status in terms of selected metric such as delay.

Load-balancing is needed to effectively use available sources and keep the nodes energy consumption balanced by equally distributing the load. The problem is to route data packets avoiding congested path so as to balance traffic load over network and lower end-to-end delay. Distributing the load within the network has two advantages. First, resource of the network is fully utilized through distributing network load. An efficient load-balancing routing protocol is able to improve packet delivery rate and network throughput. Second, energy consumption is balanced by equally distributed load, so that the network lifetime could be prolonged. A dynamic parameterless load-balancing georouting protocols was proposed in [Sp]. The node holding the packet for delivery compares costs of sending the packet to all available neighbors that are closer to destination and not fully loaded, against the progress made. The neighbor with the minimum cost over progress ratio \( \text{Load}(A)/(|SD|-|AD|) \) is selected. In this formulation, the load (as selected cost) could be the ratio of already consumed bandwidth over total bandwidth at node \( A \). The cost is then increasing linearly with the consumed bandwidth. A more progressive cost can be used by defining \( \text{Load}(A) = 1/capacity(A) \), where \( \text{capacity}(A) \) is the normalized remaining bandwidth (capacity) at neighboring node \( A \) [Sp].

### 4.4. Memorization based georouting with guaranteed delivery

Greedy based routing stops if a current node can not find any neighbor that is making an advance with respect to selected advance mechanism, such as reducing distance to the destination. However, a route from source to destination may still exist. We consider only localized methods to find route when selected greedy technique fails. They are generally divided into two classes, depending on whether or not any information about the route has been left at visited nodes, for possible later consultation. This is not allowed in memoryless routing, where all needed information is included in the packet. We will first cover some techniques that do allow memorization. In some cases, memorization can be justified. One example is the creation of a path between two nodes that will be used for an ongoing traffic between them (e.g. for Quality of Service based applications), where nodes on the path need to memorize the next hop. The alternative is obviously to record the whole path in the message, but with increased path lengths this method does not scale well, as increased message size increases collisions and reduced bandwidth. We will describe several recovery mechanisms based on memorization.

Stojmenovic and Lin [SL01a] proposed flooding based methods, called \( f\text{-greedy} \) and \( f\text{-MFR} \), which apply greedy routing and MFR at intermediate nodes and run a recovery mechanism at concave nodes. Each concave node memorizes message IDs and rejects further copies of the same message. That is, neighbors of concave node \( C \) learn about \( C \)’s concave status from the packet and do not select \( C \) as next hop forwarding node in future
attempts. Each neighbor of the concave node initiates a separate routing task toward the destination, in parallel. Some paths can be terminated later when reaching a node already handling same routing task for a previous path via that node.

A localized DFS (Depth First Search)-based routing algorithm was proposed by Stojmenovic, Russell and Vukojevic [SRV02]. Different from f-greedy, DFS is single path routing. Each node remembers if it has already been visited by the DFS traversal, and the node from where the message was received for the first time. Packet forwarding is performed by sorting all neighbor nodes according to their distance from the destination $D$ and selecting the node that is the closest to $D$. Since already-visited nodes have already transmitted a forward packet which is overheard by their neighbors, the neighbors can learn their status and do not select them for forwarding again [VGKNS08]. A returned message will be sent to the next choice in the sorted list of all next hop nodes. If all neighbors are explored and return the message, the message is returned to the node from which it was first received. DFS method [VGKNS08] is applied to arbitrary cost metric. Neighbors of current node are sorted based on the cost-to-progress ratio that they can provide, and used in that order in attempts to find a route.

Fig. 4.5 DFS routing before ($SBFMFMNFGHLKJD$) and after ($SBFMNFGHLKJD$) enhancements.

In the example in Fig. 4.5, according the DFS algorithm in [SRV02], source $S$ sorts three neighbors according to the distance to $D$, and then forwards to $B$ which is closest to $D$. Similarly, $B$ sorts its neighbors $A$, $C$ and $F$, excluding sender $S$, and forwards to $F$. $F$ forwards to $N$ which then forwards to $M$. $M$ forwards to $F$ which has been visited. Thus, $F$ rejects the message to $M$. $M$ returns to $N$ since it has no available neighbors. Similarly, $N$ returns to $F$. $F$ does not forwards to its second neighbor $M$ since it received forwarding message from $M$ already. Instead, $F$ forwards to its last neighbor $G$. $G$ forwards to $H$ which ultimately reaches $D$ via $HLKJD$. The routing path of DFS before enhancement is $SBFMFMNFGHLKJD$. In the enhanced version of DFS [VGKNS08], nodes can learn their neighbors’ status by overhearing the transmissions of already-visited neighbors. In the example, $M$ eliminates both $F$ and $N$ from its available neighbor list after $F$ forwards
to $N$. $F$ eliminates both $N$ and $M$ from its available neighbor list after $N$ forwards to $M$. So the routing path of DFS after enhancement is \textit{SFBNMNFGHLKJD}.

The method, however, does not work well in ‘island’ areas with dense node population, as all these nodes would be visited before exiting the island to try another route toward destination. To reduce this problem, DFS is applied only on nodes from a connected dominating set [VGKNS08].

Application of DFS routing for QoS support was further discussed in [SRV02]. Based on the information on its own physical location and periodically updated location information of all neighbors, each node is able to estimate the current speed of any neighbor and estimate how long the link will exist. The information could be used to find a route which supports a specified connection-time requirement. In addition, a minimum bandwidth requirement and maximum delay may be considered as well during DFS traversal. A message is returned once the maximum delay is exceeded or no outgoing edge matches the minimum bandwidth requirement. Intermediate nodes along the path memorize the uplink and downlink of the path, such that the QoS communication between the source and the destination can be set up.

Ma, Sun, Zhao, and Liu [MSZL08] proposed a detouring mode for any geographical routing protocol (but it has no impact on greedy routing). The strategy is applied to prune the path found by georouting protocol when the first packet is routed. After the first packet is delivered, a pruned path is also obtained, and subsequent packets can be forwarded using the pruned path. Suppose that a node $A$ forwards a packet to its neighbor $B$, and afterwards hears the same packet being forwarded by another neighbor $C$, node $A$ can immediately make a shortcut by forwarding other packets for same destination to $C$ directly, bypassing at least node $B$. The algorithm requires some state information to be recorded at nodes that make shortcuts. Similar bypassing algorithm was previously applied for the specific case of depth first search (DFS) routing algorithms in [SRV02], as natural part of DFS process, when $C=A$, as part of constructing QoS route out of initial route. The bypassing algorithm can be extended to allow common neighbors to intervene in the path. Suppose that node $B$ hears a packet being forwarded by its neighbor $A$, and later on by its neighbor $C$, with hop count being increased by more than 2. Then node $B$ may offer node $A$ to forward future packets, which will be then delivered to neighbor $C$, thus making detour with two hops from $A$ to $C$.

4.5. Guaranteed Delivery without Memorization

4.5.1. Face routing in planar geometric graphs

A planar graph is the graph which can be drawn on the plane in such a way that edges intersect only at their endpoints. Fig. 4.6 shows a planar graph. In geometric graphs, each node is aware of positions of itself and its neighbors, and therefore angles toward them. Examples of planar and geometric graphs include street maps, or rooms on the same floor in a building. Planar geometric graphs divide graph into faces. In the example in Fig. 4.6, face $F_1$ is polygon $SABC$ and face $F_2$ is polygon $BEFGHC$. Kranakis, Singh and Urrutia [KSU99] described the first localized memoryless routing algorithm for planar geometric graphs, which guarantees delivery whenever source and destination are connected.
The Face routing in [BMSU99] is an improvement on the routing algorithm in [KSU99]. The main idea of the face routing [BMSU99] is to advance (toward destination $D$) intersections of faces with a straight line segment that connects the last intersection $X$ (initially it is the source $S$) and the destination $D$. A packet is routed along the interiors of the faces until an edge on the route intersects $XD$ between $X$ and $D$. In Fig. 4.6, the line intersects the planar graph with faces $F_1$, $F_2$, ..., $F_7$. The boundary of any face can be traversed by applying the right-hand rule (counterclockwise traversal) or the left-hand rule (clockwise traversal). In the right-hand rule, face is traversed by keeping the right hand on the wall while walking forward. That is, the packet is forwarded along the next edge counterclockwise from the edge where it arrived. When the packet arrives at an edge intersecting the line $XD$, the next face intersected by the line is handled in the same way. The process continues until the packet reaches the destination $D$ or the first edge of current face traversal is traversed twice in the same direction (this case indicates that source and destination are disconnected, and a loop is created).

Consider example in Fig. 4.6, and assume left-hand rule is applied. Face $F_1$ intersects line $SD$ and the packet traverses in $F_1$ over path $SABC$ until edge $BC$ intersects line $SD$ at point $X_1$. The next face $F_2$ is traversed on path $CBE$ until the next intersection $X_2$, followed by face $F_3$ on path $EBFE$ and intersection $X_3$. Path then follows face $F_2$ again along $EFG$, $F_4$ along $GFI$, $F_5$ along $IFONMLK$ and finally face $F_6$ until delivery to $D$. The whole path is indicated with dashed edges. This variant normally traverses intersecting edges back and forth and is also known as after crossing variant, and can be applied similarly with the right-hand rule on each face. If we want to avoid intersecting edges twice, a before crossing variant can be used. It can also start with the left or right hand-rule, or choice among them may be based on some other criteria, for example using smaller initial angle/direction toward $D$. Face $F_1$ is then traversed by edge $SC$ which has closer direction to $SD$ than edge $SA$. The next face $F_2$ is selected and the reference line is updated to $X_1D$. The packet is forwarded from $C$ to $H$ and then to $G$ until edge $GF$ intersects line $X_1D$ at point $X_4$. Similarly, the face switches to $F_4$ and the packet is forwarded from $G$ to $I$ with edge $IF$ intersecting line $X_4D$ at point $X_5$. Finally, the packet will reach $D$ via path $SCHGJKD$ in bold line.

Note that the traversed face after the intersection may be the same face traversed before the intersection with $XD$. Suppose the destination is point $P$ on line $SD$ in Fig. 4.6,
and suppose that the same after crossing variant (dashed line) is followed. While traversing $F_5$, at point $K$, path will continue on $F_5$ along $KJIQ$. At intersection $X_7$ of $IQ$ and $SP$, face $F_5$ is selected again and is traversed by algorithm [BMSU99] since it contains the line $X_7P$, and message is delivered to $P$ along $IQP$ in the after crossing or left-hand variants, or along $X_5JKLMNOFX_7RQP$ line (not shown in the image) in the before crossing or right-hand variants. The algorithm [KK00] however forces the change of face at every intersection $X_iD$ (note that the $X_i$ must be internal point on the line segment $X_{i-1}D$). Then [KK00] selects face $F_7$, and the message traverses along $QIR$ or $IRQ$ indefinitely in a loop since $X_6$ is not inside the line segment $X_7P$. The face change will never occur, resulting in a loop. Thus this step is correctly implemented in [BMSU99], while it has been mistakenly described in [KK00], leading to lack of guaranteed delivery in arbitrary planar geometric graphs.

There are several variants of face routing, mainly addressing various decisions on the choice of left or right hand rules to use for each traversed face. The differences can also be made based on changing face before crossing an edge on the route, or after crossing it. Few variants of the face algorithm can be described as follows.

**Face routing**

// $S$: source, $D$: destination

$X \leftarrow S$

repeat

let $f$ be the face of $G$ with $X$ on its boundary that intersects line $XD$

traverse $f$ (counter)clockwise until reaching an edge $UV$

that intersects line segment ($X, D$) at some internal point $Q \neq X$

$X \leftarrow Q$

continue routing from node $U$ (before crossing) or $V$ (after crossing $XD$)

until $X = D$.

### 4.5.2. Gabriel Graph

We are interested in face routing in UDGs, not in planar geometric graphs. This is because UDG (Unit disk graph) is utilized to model wireless ad hoc and sensor networks. In general, an arbitrary unit disk graph is not planar. We therefore need a geometric structure that will be derived from UDG and will provide planar graph in localized manner. Currently, the most convenient structure for routing applications is Gabriel graph (GG).

![Fig. 4.7 Forbidden region for (a) Gabriel graph (b) Delaunay triangulation.](image-url)
Gabriel graph was first introduced by in [GS69]. Two points $u$ and $v$ are joined by an edge in the GG whenever the disk with diameter $|uv|$ contains no other points from the given point set. In Fig. 4.7 (a), the dashed circle is the forbidden region of GG where there is no other node. Fig. 4.8 shows a GG constructed from an UDG. Bold edges belong to GG and all edges belong to UDG. In the 3D definition of Gabriel graph, two points are joined if the sphere with them as endpoints of its diameter does not contain other points from the set.

GG concept is applied on the top of an UDG. Let $S$ be a set of points in the plane and $U(S)$ is the unit disk graph which contains all points in $S$. Let $GG(S)$ denote the Gabriel graph induced by $S$. The following theorem shows the Gabriel graph preserves the connectivity of UDG.

**Theorem 4.1** [BMSU99] If $U(S)$ is connected, then $GG(S) \cap U(S)$ is connected.

**Proof.** We will prove that both graphs contain minimal spanning tree $MST(S)$ of the same set $S$, therefore their intersection also contains $MST(S)$, and is therefore a connected graph. We will prove $MST(S) \subseteq GG(S)$ by contradiction. If $MST(S)$ is not subset of GG then there exists edge $PQ \in MST(S)$, $PQ \notin GG$. Since $PQ$ is not in GG, there exists node $W$ inside disk with diameter $PQ$. This node $W$ then satisfies $PW < PQ$, $QW < PQ$. Because $PQ$ is in $MST(S)$, either $PW$ or $QW$ is not in MST. Assume $PW \notin MST(S)$. Replace then $PQ$ by $PW$ in $MST(S)$. The new $MST(S)$ has smaller sum of edge lengths, which is a contradiction. Therefore $GG(S)$ contains $MST(S)$, and is therefore connected. Suppose that the radius of $U(S)$ is $R$. All edges of $U(S)$ are thus not greater than $R$. According to Kruskal’s algorithm [K56], these edges are first considered in constructing $MST(S)$ before other edges whose length is greater than $R$. After considering of all edges in connected $U(S)$, $MST(S)$ includes all nodes from $S$ and is already completed. That is, $MST(S) \subseteq U(S)$. So, $GG(S) \cap U(S)$ is connected. ♦

**Theorem 4.2.** Gabriel graph is a planar graph.

**Proof.** Proof is by contradiction. Assume that it is not a planar graph. Then there exist two intersecting edges $UV$ and $PQ$. From $UV$, $PQ \in GG(S)$, it follows that $\angle PUQ < \pi/2$, $\angle PVQ < \pi/2$, $\angle UPV < \pi/2$, $\angle UQV < \pi/2$. Then the sum of angles in quadrilateral $UPVQ$ is $< 2\pi$, which is a contradiction. ♦
The construction of GG is straightforward from its definition. To test whether or not an edge $uv$ belongs to GG, node $u$ can check if distance from other points to the center of line segment $UV$ is $>|UV|/2$. Alternatively, $u$ can verify if angles over $uv$ from each of neighboring points is acute. If so, the edge is in GG, otherwise it is not included in GG.

The computational time complexity of this algorithm for testing all edges at a node is $O(d^2)$, where $d$ is the maximum degree in the network. However, the communication in wireless networks is much more expensive than computation. If node $u$ is already aware of the geographic locations of itself and all its neighbors, no additional messages are involved in the construction of GG. This makes GG a localized scheme. Moreover, it is a zero message planar graph construction method, a very desirable property.

RNG graph is described in Chapter 2. It has been pointed out [LWW01] that the number of edges in RNG and GG are bounded by $3n-10$ and $3n-8$, respectively, where $n$ is the number of nodes in the networks. Thus the average node degree in RNG and GG is bounded by 6. In fact, the average degree of RNG is about 2.5 [HLS05] while the average degree of GG is about 3.8 [HLR04]. The maximum node degree of each node of RNG can be reduced to 5 if length of each edge is made unique. Suppose a node $u$ in RNG has degree greater than 5. There must exist two neighbors $v$ and $w$, such that $\angle uvw \leq 60^\circ$. Since $|uv| \neq |uw|$ (suppose $|uv|>|uw|$), node $w$ must be inside the forbidden area of $uv$, contradicting that $uv$ is in RNG. To make all edges unique, it suffices to consider them as records with primary, secondary and ternary keys being $(|uv|, \min(u,v), \max(u,v))$.

We will also briefly discuss Delaunay triangulations (DT) as possible planar graph alternatives to GG. An edge $uv$ is in DT if there exists a circle, whose chord is $uv$, which does not contain any other node in its interior. Alternatively, DT contains all triangles $uvw$ which satisfy that the circle passing through $u$, $v$ and $w$ does not contain any other node (see Fig. 4.7 (b)). However, DT cannot be constructed by localized algorithms. The reason is that neighbor information of a node alone is not sufficient to determine if any triangle in which the node is an endpoint belongs to DT. Node $u$ may not be aware of existence of node $x$ if $x$ is inside the circle but outside the $u$’s communication area. That is, circle sizes in the definitions are not limited, and localized knowledge then is insufficient.

From the first definition of DT, we can see that GG $\subseteq$ DT. Suppose that an edge $uv$ belongs to RNG. This means that there is no node $w$ such that $|uv|>|uw|$ and $|uv|>|vw|$. Thus $w$ is not inside circle with diameter $uv$. That is, RNG $\subseteq$ GG. Therefore, we have MST $\subseteq$ RNG $\subseteq$ GG $\subseteq$ DT [HLS05].

4.5.3. Routing with Gabriel graph

Gabriel graph can serve as a planar graph needed for face routing to work. The main problem with respect to the performance of face routing is exploring significant portions of boundaries of faces. Therefore, Bose, Morin, Stojmenovic and Urrutia [BMSU99] proposed a combination of the face routing algorithm with the distance-based greedy routing. The algorithm, that is referred to as GFG (greedy-face-greedy), applies greedy algorithm until the packet reaches a node such that all its neighbors are further from the destination than the node itself. The face routing is applied until the packet reaches another node that is strictly closer to the destination. The greedy algorithm is then resumed. The algorithm can switch between greedy and face mode several times, but guarantees progress and delivery because face routing is always successful, and loop
cannot be created since the algorithm always advances in greedy mode, and is guaranteed to further advance while in face mode (that is, it is guaranteed to recover).

![Figure 4.9 Traversing the face until recovery.](image)

The example in Fig. 4.9 illustrates how the face routing can route over the void area. Suppose the greedy algorithm is applied and the packet reaches node $S$ which has no neighbors closer to destination $D$ than itself. Face routing is required for recovery, i.e. to find a node that is strictly closer to $D$ than $S$. If the right-hand rule is applied, the packet follows the shorter route until it reaches the first node $B$, which is closer to $D$ than node $S$. Similarly, the packet follows the longer route and reaches node $C$ if the left-hand rule is applied. The greedy algorithm is then resumed after recovery.

GFG is illustrated in Fig. 4.10, from source $S$ to destination $D$. Bold edges belong to GG and all edges belong to UDG. Since $S$ cannot find any neighbor which is closer to $D$ than itself, face routing is applied first. Suppose the left-hand rule is used, the packet follows route $SYXTPOK$ until it reaches the first node $K$ with $|KD|<|SD|$. The greedy algorithm is then applied and the packet advances following the route $KGHI$ until it reaches node $I$ which has no neighbors closer to $D$ than $I$. The face routing is applied again and the packet follows route $IHGFECB$ until it reaches node $B$ with $|BD|<|ID|$. Note that face routing considers only edges in GG and edge $BE$ is not selected. The greedy algorithm is then applied and the packet finally reaches $D$ via the route $BAD$. If the right-hand rule is applied, the packet follows the route $S\rightarrow Z\rightarrow S\rightarrow Y\rightarrow X\rightarrow W\rightarrow V\rightarrow W\rightarrow U\rightarrow T\rightarrow R\rightarrow T\rightarrow P\rightarrow Q\rightarrow P\rightarrow O\rightarrow M\rightarrow N\rightarrow M\rightarrow L\rightarrow J\rightarrow F$ and reaches the first node $F$ with $|FD|<|SD|$. The greedy algorithm is then applied and the packet follows the route $FEBAD$ (edge $BE$ is selected in the greedy routing) until it reaches the destination $D$. 

14
Fig. 4.10 GFG routing from $S$ to $D$ by right-hand and left-hand recoveries.

If source is disconnected from destination, GFG will create a loop. A loop is detected if the first edge in face routing is repeated twice by the traversal, in the same direction. The first edge can be added to the message, so that its repetition can be detected. Note that repeating any node on the path is not sufficient to declare a loop.

The variant of GFG using GG as planar graph is expected to have smaller average hop count than variant with RNG in face routing because GG is denser than RNG (RNG is a subset of GG), and therefore has more edges. More edges leads to fewer edges on faces and therefore to fewer hops.

The GFG algorithm was further improved by Datta, Stojmenovic and Wu [DSW02] to reduce its average hop count. Each forwarding node uses the local 2-hop information available to calculate as many hops as possible and forwards the message to the last known hop directly instead of forwarding it to the next hop. Connected Dominating Sets (CDS) technique was further applied and face routing was performed on the dominating set except possibly the first and the last hops.

GFG algorithm with added IEEE 802.11 medium access layer was later implemented as the greedy perimeter stateless routing (GPSR) protocol by Karp and Kung [KK00]. The GPSR protocol is a variation of GFG. More precisely, GPSR uses the before-crossing instead of after-crossing variant, and discusses the relative neighborhood graph (RNG) as an alternative to the GG. However, these modifications do not improve the performance of the routing protocol. It was pointed out [KGKS05] that GPSR can not guarantee delivery in arbitrary planar graphs. The fact was confirmed by Frey and Stojmenovic [FS06] and formal proof of delivery guarantee of GFG in arbitrary planar graphs was further given. This is due to a difference in the face routing procedure between GFG and GPSR. GPSR always switches to the other face whenever the line $XD$ in algorithm is intersected. However, GFG correctly merely selected the proper face, which sometimes can be very same face before the intersection, as illustrated in Fig. 4.6, once an intersection point is found. Interestingly, face switch does not occur at all when
GG is used as planar graph, which is the reason why the error in GPSR implementation was not identified before reporting it in [FS06]. More detailed difference between GFG and GPSR could be found in [FS06]. The following theorem shows that under Gabriel graph, when recovering from a greedy routing failure, it is always possible to reach a node which is closer to the destination than the current node. So, the greedy algorithm can be resumed after traversing only one face with face routing.

**Theorem 4.2** [FS06] Let $SD$ be the line between source node $S$ and destination node $D$ in a Gabriel graph $G$. For any edge $UV$ in GG intersecting the line $SD$, the distance between $D$ and at least one of the edge end points $U$ or $V$ is smaller than the distance between $S$ and $D$.

**Proof.** Since edge $UV$ is in Gabriel graph, the circle having $|UV|$ as its diameter does not contain nodes $S$ and $D$. Thus, both angles $\angle USV$ and $\angle UTV$ are less than $90^\circ$ (see Fig. 4.11). Since the angles of the quadrilateral $SUDV$ sum up to $360^\circ$, at least one of the angles $\angle SUD$ and $\angle SVD$ is greater than $90^\circ$. This makes $SD$ the longest edges in corresponding triangle. Therefore, at least one of the two nodes $U$ or $V$ is located closer to $D$ than the node $S$. ♦

Therefore, the GFG that uses GG can be implemented by the following simplified algorithm.

**GFG over GG** [FS06]

```
repeat
  follow greedy until delivery or failure at node $S$
  if failure at $S$ then
    select face $f$ containing the line $SD$
    traverse $f$ until return to greedy is possible
  endif
until delivery
```

Kuhn, Wattenhofer, Zhang and Zollinger [KWZZ03] proposed an extension of the GFG algorithm which is referred as **greedy other adaptive face routing plus** (GOAFR+). It observes that efficiency of face routing depends on the traversal direction of a face. An
improper traversal direction of a face may result in a long path to the destination. So, the basic idea of GOAFR+ is to introduce a circle $C$ centered at the destination and its initial radius is set to include the source. The greedy mode is employed as long as there is a next hop node closer to the destination, and whenever possible the radius of $C$ is exponentially decreased as long as the current visited node is within $C$. Once greedy mode reaches a concave node $U$, a modified version of face routing is invoked. If the face is traversed completely without hitting the circle $C$, the packet is sent to the node visited so far which is closer to the destination than $U$ (if no node is closer to the destination than $U$, routing failure will be reported.). If $C$ is hit for the first time, the face is traversed in the opposite direction. If $C$ is hit for the second time and none of visited nodes is closer to the destination than $U$, face traversal continues as if started at $U$ and radius of $C$ is exponentially increased. Once face traversal visits up to a predetermined constant factor more nodes closer to the destination, GOAFR+ interrupts face traversal and switches to greedy mode again.

The cost of a path is defined as the sum of costs of all edges in the path, where the cost of an edge could be any cost metric which is polynomial in the Euclidean distance. In [KWZ02] it has been shown that the worst case cost of any geometric routing algorithm is bounded by at most quadratic path costs (compared to the shortest weighted path), which is denoted as asymptotic optimality. It was pointed out [KWZ03] that asymptotic optimality cannot be achieved if face traversal is switching back to greedy mode when the line that connects the concave node and the destination is intersected for the first time (e.g., GFG is not asymptotically optimal). The combination of greedy and face routing becomes asymptotically optimal when packets explore the complete face and switch back to greedy mode at the face edge which is closest to the destination. It has been proven [KWZ03] that GOAFR+ is asymptotic optimal although it does not traverse the complete face in general.

There are significant challenges left on georouting with guaranteed delivery, and hundreds of papers in literature address them. Imprecise location information is a significant challenge for georouting with guaranteed delivery. There is no localized memoryless algorithm for georouting in 3D that has guaranteed delivery property. Unit disk graph with equal transmission radii and absence of obstacles are required for Gabriel graph to remain connected. An extension for fuzzy unit disk graphs is given in [BFNO01]. Two nodes are connected if their distance is smaller than $r$ and are disconnected if the distance is greater than $R$. Two nodes are randomly connected or disconnected at the distance between $r$ and $R$. The main algorithm works if $R/r < 1.41$, with certain extensions and message overhead to maintain GG. For other cost metrics, there is still no real alternative to GG based face routing for recovery mode, which prefers close neighbors. Existing improvements are based on shortcuts, dominating sets, and shortest weighted paths over face traversed edges.

One of challenging problems is also addressing mobility issues for intermediate nodes on routes, while routing is in progress. The algorithm is loop free for static networks, but loops can be created by mobile nodes. After entering a face, two nodes on the same face can move close to each other and divide the face into two new faces, leaving message in one of faces that does not intersect imaginary line from source to destination, thus looping forever. However, this problem can be partially resolved (if mobility is not so
high that required information becomes unreachable) by adding timestamp of the last intersection with imaginary line SD and ignoring links created afterwards.

### 4.6 Beaconless Georouting

The greedy forwarding algorithms normally need to periodically exchange ‘hello’ messages (beaconing) with maximum signal strength by each node in order to broadcast current position information to all one-hop neighbors. The beaconing process of greedy routing costs additional energy consumption which occurs independently of current data traffic.

Heissenbuttel and Braun proposed the *beaconless routing* (BLR) algorithm in [HB04]. The BLR was further integrated with the IEEE 802.11 MAC layer in the *contention-based forwarding* (CBF) by Füßler et al. [FWKMH03] and *implicit geographic forwarding* (IGF) by Blum et al. [BHSS03]. In BLR, node $S$ currently holding the packet destined for node $D$ will include its own and location of $D$ in the packet, and retransmit either only request for forwarding, or full message content. Upon receiving the packet, neighboring node, candidate for forwarding with a progress, calculates a waiting time-out depending on the relative location coordinates of itself, $S$ and $D$. The node located at the “best” location introduces the shortest delays and forwards the packet first (or responds first with the offer to retransmit). The (most) remaining nodes then cancel the scheduled transmission of the same packet.

**Fig. 4.12 Forwarding area in BLR.**

BLR is illustrated in Fig. 4.12. To ensure that all potential forwarding nodes detect transmission of $S$, selection of candidate nodes for the next forwarding step is limited in a certain forwarding area. The forwarding area has the property that each node is able to overhear the transmission of any other node in the area. Heissenbuttel and Braun [HB04] showed that the circle with a diameter equal to the transmission radius, centered at the line $SD$ with $S$ as one endpoint (the dotted circle in Fig. 4.12) is a good forwarding area with regard to progress and successful hops before greedy routing fails. Several delay functions are investigated, resulting in different forwarding behavior.

A technique called the *active selection method* was further proposed in [FWKMH03]. A forwarding node sends a control packet instead of the full message to all its neighbors. Neighbors respond with information of their forward progress after a time-out which depends on their distance to the destination. The forwarding node then sends the full message that indicates which of its neighbors will forward the message. Zorzi [Z04] proposed to avoid duplicate forwarding in a BLR scheme by employing the request-to-
send/clear-to-send (RTS/CTS) MAC scheme from IEEE 802.11. The current node sends an RTS signal instead of the message and waits for a CTS signal. If several responses are received, the node selects the one that appears to be the best for forwarding and then sends the message to that neighbor directly.

When greedy routing fails, a recovery strategy such as face routing is required to guarantee delivery. Note that face routing is normally based on a planar subgraph which is constructed from neighborhood information, which is not available in beaconless routing. To solve the problem, Kalosha, Nayak, Ruhrup, and Stojmenovic [KNRS08] proposed beaconless georouting schemes with guaranteed delivery. They proposed two solutions: Beaconless Forwarder Planarization (BFP) and Angular Relaying. BFP finds correct edges of a local planar subgraph at the forwarder node without hearing from all neighbors. The face routing is then applied in the subgraph. Angular Relaying directly determines the next hop of a face traversal. Details of the both schemes are presented next.

BFP is a general scheme which is used to construct Gabriel graph (GG) and Relative Neighborhood Graph (RNG). It consists of two phases: selection phase and protest phase. In the selection phase, the forwarder \( v \) broadcasts an RTS including its own location and sets its timer to \( t_{\text{max}} \). Each neighbor \( u \) sets its contention timer by using the following delay function:

\[
t(d) = t_{\text{max}} \times d/r,
\]

where \( d = |uv| \), \( r \) is the transmission radius and \( t_{\text{max}} \) is the maximum timeout. That is, the closer neighbors set smaller waiting times. A node responds with a CTS when its contention timer expires. If a node \( w \) receives the CTS of another node \( w' \) that lies in the forbidden region \( N(v, w) \), \( w \) cancels its timer. In the example in Fig. 4.13, nodes set their timers in increasing order \( w_1, w_2, \ldots, w_6 \) according to their distance to node \( v \). Node \( w_1 \) responds first. When contention timer of \( w_2 \) expires, \( w_2 \) responds with a CTS including its location to \( v \). Since \( w_5 \) overhears the CTS, it finds \( w_2 \) is in the forbidden region \( N(v, w_5) \). Thus, \( w_5 \) cancels its timer and is referred to as a hidden node. Since \( w_6 \) does not hear the CTS from both \( w_4 \) and \( w_5 \), it will respond a CTS once its contention timer expires. However, edge \( vw_6 \) is the violating edge and should not be included in the GG since \( w_4 \) and \( w_5 \) are in the forbidden region \( N(v, w_6) \).

![Fig. 4.13 Selecting GG edges in BFP.](image)

In the protest phase, hidden nodes protest against violating edges. If the set of violating nodes is not empty, the hidden node starts its timer by using the same delay function. Violating node could be reported by some other hidden node. Otherwise, the hidden node sends the protest message. Upon receiving protests from hidden nodes, the forwarder removes violating edges and finally obtains a planar graph. In the example in Fig. 4.13, timer of hidden node \( w_4 \) is smaller than the timer of \( w_5 \). When timer of \( w_4 \) expires, it sends a protest to the forwarder \( v \). \( v \) removes violating edge \( vw_6 \). Since \( w_5 \)
overhears the protest of \( w_4 \), it removes \( w_6 \) from the set of violating nodes which then becomes empty. Thus, \( w_5 \) remains silent after its timer expires.

Similar to BFP, the Angular Relaying algorithm consists of selection and protest phases. In the selection phase, the forwarder node \( v \) that receives a packet from the previous hop \( u \) sends an RTS including location of \( u \) and itself, and sets its timer to \( t_{\text{max}} \). Each neighbor, say \( w \), sets its contention timer by using following delay function:

\[
{t(\theta)} = t_{\text{max}} \times \theta \times (2\pi),
\]

where \( \theta = \angle uvw \) is in counter-clockwise order. A neighbor responds by an “invalid CTS” if it finds other nodes in the forbidden region. The purpose is to let other nodes be aware of its existence. Otherwise they would be hidden and need a chance to protest later. Once the first candidate \( w \) answers with a valid CTS, the forwarder immediately sends a SELECT message announcing \( w \) is the first selected node. All candidates with pending CTS answers cancel their timers. The protest phase starts once the first candidate is selected. The forwarder sets its protest timer that covers the time when protests can occur (e.g., \( t(\pi/2) \) for GG). No further CTS is allowed. Each candidate node \( x \) sets a new timer \( t(\theta) \) which determines the order of protests where \( \theta = \angle uvx \) – \( \angle uvw \). Only nodes in \( N(v, w) \) are allowed to protest. Node \( x \) that protests automatically becomes the next hop. Afterwards, only nodes in \( N(v, x) \) are allowed to protest. Finally, the forwarder sends the data packet to the currently selected candidate after its timer expires.

![Diagram](image)

**Fig. 4.14 Selecting the next hop \( w_5 \) after invalid CTS from \( w_1, w_2, w_3, w_4 \) in Angular Relaying.**

In the example in Fig. 4.14, nodes set their timer in increasing order, \( \ldots \), \( w_6 \). Since \( w_1 \) is in region \( B \) (the previous hop \( u \) is in \( N(v, w_1) \)), \( w_1 \) sends invalid CTS. Similarly, \( w_2 \) sends invalid CTS since \( w_1 \) is in \( N(v, w_2) \). After \( w_3 \) sends a valid CTS, it is the selected node. However, \( w_4 \) protests, and believes it is the next hop. The protest then is sent by \( w_5 \). Finally, \( w_5 \) is the next hop and \( v \) sends the data packet to \( w_5 \) directly.

### 4.7 Georouting with virtual and tree coordinates

The accuracy of exact geographic coordinates that is currently available is not sufficient to support the claimed performance of georouting algorithms. An alternative solution is to use virtual coordinates instead of real ones. Typical approach is to assign hop count distances to certain set of landmark nodes as coordinates of sensor nodes, and define distance between nodes as the sum of (absolute values of) differences in hop count toward these landmarks (this is also called Hamming distance). Greedy routing can be applied in these coordinates [CCDU05]. The problem, however, is that different nodes can have same coordinates. The sensors are not properly sorted in such coordinates, and greedy routing may lead to a node that is local minima by Hamming distance. A
resolution is proposed in [CMT07] by adding tree coordinates to nodes, rooted from a landmark node. Greedy routing then proceeds by considering only neighbors that provide progress in tree coordinates. The combined set of two virtual coordinates then enables routing with guaranteed delivery.

Mitton, Razafindralambo, Simplot-Ryl, and Stojmenovic [MRSS] consider the problem of designing power efficient routing with guaranteed delivery for sensor networks with known distances between neighbors but unknown geographic locations. They propose HECTOR, a hybrid energy efficient tree-based optimized routing protocol, based on two sets of virtual coordinates. One set is based on rooted tree coordinates, and the other is based on hop distances toward several landmarks. In the algorithm, the node currently holding the packet will forward it to its neighbor that optimizes ratio of power cost over distance progress with landmark coordinates, among nodes that reduce landmark coordinates and do not increase tree coordinates. If such a node does not exist then forwarding is made to the neighbor that reduces tree based distance and optimizes power cost over tree distance progress ratio.

4.8. Georouting in Sensor and Actuator Networks

Geographic routing by an adaptive targeting (RAT) protocol was studied by Shah, Bozyiqit and Aksoy [SBA07]. RAT provides sensor-to-actuator communication and dynamic coordination of actuators in response to emergencies. It focuses on applications in which time critical data is required to be sent from sensor nodes to actuators. A sensor node is said to be covered by an actuator if the distance between them is not greater than R. Sensors report only to a single actuator. Actuators broadcast subscribe messages to sensors in their covered areas. When they move, such message is sent with frequency based on speed and field dimensions. Actuator-actuator coordination is by broadcasting whose details are not given.

RAT consists of two components: Delay-constrained geographical-based routing (DC-GEO) and Integrated Pull/Push (IPP) coordination. In IPP, actuator nodes subscribe to specific events of their interest in the field and sensor nodes disseminate the event readings to subscribed actuators for time periods of subscription life. Sensor nodes push the data as long as there is any actuator interested for the observed event. DC-GEO employs greedy forwarding such that delay constraint can be met as well as energy consumption of forwarding nodes is balanced.

The process of forwarding the data packets toward actuators consists of two steps. The source node sets the time to live (TTL) field in the packet and each forwarding node updates the TTL by deducting the traversed hop delay. Each sensor constructs a delay-constrained forwarding subset (DCFS) from the neighbors which are closer to the destination, such that the given delay constraint can be met. The second step of forwarding is to balance the load of nodes in DCFS by selecting the forwarding node which has the highest energy level. Proposed routing protocol minimizes energy while meeting delay constraint. Current node, say i, first constructs DCFS from its neighbors for each actuator a, such that the given delay constraint can be met. It applies the criterion \(\frac{\text{TTL}}{T(j)} > \frac{D(i, a)}{D(i, j)}\) on each of its neighbor j, where TTL represents the constraint of remaining time to reach destination, \(T(j)\) is the expected delay to relay a packet from i to j, \(D(i, a)\) and \(D(i, j)\) are distances from current node i to actuator a, and candidate neighbor j, respectively. From DCFS, i selects the next node which has the highest
residual energy. Actuators need to be positioned close to emergency area to reduce response time.

When no acceptable neighbor exists (there might be greedy neighbors providing advance but not meeting delay criterion), routing continues in face mode, by variant of GFG [BMSU99] that follows both faces. The concave node relays the packet to one node on the left and one node on the right along the perimeter of the current face. Then each of these receivers continues with face routing until the next recovery, where algorithm can return to greedy mode based on delay bounds. Each of future concave nodes on any branch may further branch in two. Authors claimed that one of branches may loop and denote then failure of that branch. However this does not follow from GFG algorithm, which requests a link to be repeated in the same direction, not that merely a node repeats. The algorithm then overall does not guarantee delivery when one exists, because such node may be on the way to recovery. When GFG works, response time might be improved, but could be also longer because face mode normally involves short edges, and increased number of hops increases delay. Further, one slow link in greedy mode does may still lead to a route with overall acceptable delay. Thus algorithm can be revisited.

4.9. Link quality metric in sensor and actuator networks

A resource-aware and link quality based (RLQ) routing metric for sensor and actuator networks was studied by Gungor, Sastry, Song and Integlia [GSSI07]. RLQ routing metric is a combined link cost metric which is based on both energy efficiency and link quality statistics. Energy efficiency is measured by residual energy and normalized energy cost to transmit and receive a packet. Since actuator nodes normally have higher energy resources than sensor nodes, energy cost of sensor nodes is assigned with a larger weight. Link quality is measured by the expected number of transmissions which is calculated from the packet reception rate. The neighbor with the minimum link cost is selected in packet forwarding. The RLQ [GSSI07] introduced in previous section was implemented in the Tmote Sky nodes (http://www.moteiv.com). Two radio hardware link quality metrics are used to measure the link quality during the operation of the network. They are link quality indicator (LQI) and received signal strength indicator (RSSI). RSSI is the estimate of the signal power and is calculated over 8 symbol periods, while LQI can be viewed as chip error rate and is calculated over 8 bits following the start frame delimiter. Packet reception rate represents the ratio of the number of successful packets to the total number of packets transmitted over a certain number of transmissions. The test-bed experiments show a strong correlation between the average LQI measurements and packet reception rates. It also shows a good performance of RLQ in terms of packet reception rate, network throughput and network lifetime.

Souryal and Moayeri [SM05] discuss forwarding that adapts to the time-varying channel and exploits spatial diversity to mitigate multi-path fading. The routing layer uses long term measurements of link quality (SNR) to opportunistically select next hop relays on a hop by hop basis. They find formula for packet success probability as a function of average SNR in quazi-static Raleigh fading. This is multiplied by advance toward destination for selecting the best neighbor. Routing layer will pass $M$ relay candidates to the MAC layer. If $M>1$, MAC layer pools the $M$ candidate relays for current SINR and position measurements and forwards the packet to the relay that
maximizes the expected progress for small-scale adaptivity. To avoid collisions, the relays reply in the order specified by the polling message.

4.10. Physical Layer Aspects and Case Studies of Georouting

Almost all existing literature on geographic routing employs UDG in the communication model. In UDG, two nodes can communicate with each other if and only if their distance is not greater than the common transmission radius. However, as discussed in previous chapters, UDG model is not realistic since variations of received signal strengths are not considered. It has been pointed out that impact of signal strength fluctuations sometimes is more significant than impact of node mobility [SNK05]. Therefore, reception of a packet is probabilistic. In addition to distance, the received signal strength also depends on other factors, such as environment landscape and transmission medium.

Zorzi and Armaroli [ZA03] considered advancement as a metric to be used in routing decisions. The advancement provided by a relay node is defined as the difference between the distance of the transmitting node to the intended destination, minus the distance between the relay node and the destination, multiplied by the probability of a successful transmission from the transmitting node to the relay. This idea has been later rediscovered (without citing [ZA03]) in several articles, including [KNS05] (conference version from Oct. 2004), [ZSKH08] (conference version in Nov. 2004), and [LBB05].

Kuruvila, Nayak and Stojmenovic [KNS05] [KNS06] proposed geographic routing protocols that are amenable to any realistic physical-layer model with fixed and variable packet lengths. Both cases with and without acknowledgments were considered. To employ position-based routing, the first step is to find a reasonably accurate approximation for the bit and packet reception probabilities for the given physical layer model. The lognormal shadowing model was adopted in [KNS05, KNS06]. It is represented as a function $P(q, x)$ which has approximation within 5% accuracy of the actual one, where $q$ depends on the length $L$ of the considered packet and $x$ is the distance between two nodes. The function is $P(q, x)=1-(x/R)^{\beta q/2}$ for $x<R$ and $P(q, x)=(2-x/R)^{\beta q/2}$ for $R \leq x < 2R$, and 0 otherwise, where $\beta$ is the power attenuation factor which is normally between 2 to 6 and $R$ is determined so that $P(q, x)=0.5$. Bit reception probability is $P(1, x)$ while the packet reception probability for packets with $L=120$ bits long is $P(2, x)$.

The $aEPR-u$ (expected progress routing) [KNS05], nodes send at most $u$ acknowledgements after receiving routing packets. It is to maximize the expected progress made by forwarding. In the example in Fig. 4.4, $C$ is the current node, $A$ is a neighbor of $C$ and $D$ is the destination. Let $|CD|=c$, $|AD|=a$ and $|CA|=x$. If all packets can be received successfully, $C$ will forward the packet to $A$ such that $|c-a|$ is maximized. In physical layer model, probability that $A$ receives the packet from $C$ is $p(x)$ (definition of $p(x)$ has been introduced in section 1.5 in chapter 1). The total expected hop count (number of packets) between two nodes at distance $x$ is $f(u, x)=1/[p(x)(1-(1-p(x))^u)]+u/[1-(1-p(x))^u]$ (see Chapter 1). The $aEPR-u$ selects the neighbor that maximizes $(c-a)/f(u, x)$.

The localized protocols in [KNS06] do not assume the hop-by-hop acknowledgements. It was pointed out that the packet delivery rate approaches 1 if a large number of intermediate nodes are placed between the source and the destination and distance between adjacent nodes approaching 0. Based on the observation, the $EER$ (end-
to-end routing) localized routing simply forwards the packet to neighbor $A$ that maximizes $p(x)$, i.e., the closest neighbor to $C$ among neighbors which are closer to $D$ than $C$. The process continues until the destination is reached or a node can not find neighbors closer to the destination than itself. Different from EER, the $nEPR$ (expected progress routing) algorithm forwards the packet to a neighbor which maximizes the expected progress $p(x)(c-a)$. Only the neighbors closer to the destination than the current node are considered.

The $InEPR$ (iterative expected progress routing) algorithm [KNS06] is an improved variant of $nEPR$. The algorithm operates as follows. Similar to $nEPR$, current node $C$ first finds a neighbor $A$ that maximizes $p(|CA|)(|CD|-|AD|)$. For all common neighbors of $A$ and $C$, $C$ finds the node $B$ such that $p(|CB|)<p(|BA|)>p(|CA|)$ and $p(|CB|)<p(|BA|)$ is maximized. Only the neighbors that are closer to the destination than $C$ are considered. The process repeats iteratively by checking the neighbors one by one until no improvement is possible. Node $C$ finally forwards the packet to the selected neighbor $B$ which will apply the same process for its own forwarding. This algorithm can be generalized by finding shortest weighted path $C, B_1, B_2, ..., B_n, A$ toward $A$. To apply Dijkstra shortest weighted path algorithm, logarithm of the product of probabilities is applied: $\log(p(AB_1)p(B_1B_2)\ldots p(B_nA))=\log(p(AB_1)+\ldots+\log(p(B_nA))$.

Projection Progress based algorithms [KNS06] differ from $nEPR$ schemes in the progress measure only. The progress is measured by dot product $CD·CA$ instead of $c-a$, where $CD·CA$ is the dot product of two vectors. The current node $C$ forwards the packet to a neighbor $A$ which is closer to the destination than itself, such that $p(|CA|)(CD·CA)$ is maximized. The Iterative Projection Progress scheme [KNS06] is the same as the $InEPR$ except that the projection progress method is employed to find the first candidate node.

The case of variable packet lengths on each hop and routing with hop-by-hop acknowledgments was studied by Stojmenovic, Nayak, Kuruvila [SNK05]. The localized algorithms use the expected number of transmitted bits (energy consumption) instead of the expected hop count in terms of packets in $aEPR-u$ and $InEPR$. The expected hop count $f(u, x)$ in $aEPR-u$ and $InEPR$ is replaced by the expected bit count $g(b, k)$ for routing with acknowledgments. The case of variable packet length and routing without hop-by-hop acknowledgments was also considered in ref. [SNK05].

The algorithms described so far are physical layer–based solutions for greedy position–based routing. The recovery procedure of delivery guaranteed routing [BMSU99] can be adapted to the physical layer model. Face routing is based on a planar graph in which edges are normally short. Thus, those edges in the planar graph have relatively high reception probabilities in physical layer model. Therefore, the recovery mode for the physical-layer impact routing may proceed in similarly as in UDG model. For each visited node on the face, the shortest cost path to the node can be calculated based on the cost metric which could take packet reception probability into consideration.

It was pointed out [FS05] that beaconless routing can also be adapted to the physical layer by modifying the criterion for selecting the best forwarding neighbor and the appropriate time-out. A given node announces the request for forwarding the packet several times so that the best forwarding neighbor receives it. Similarly, the best forwarding neighbor responds a few times to make sure the response is received and it is selected. The number of duplicated packets depends on the detailed physical layer model.
References


