This chapter introduces the concept of dominating sets, some important characteristics of dominating sets, the algorithm used for computing dominating sets, and application of dominating sets in routing. The concepts described in this chapter are useful in understanding the result and discussion presented later in the thesis. Dominating sets are important for finding minimal subset that provides path between every pair of nodes in a connected graph.

4.1 Dominating Sets

Dominating set based routing is a promising approach in ad hoc network to reduce the communication overheads involved in routing. The search space for a route is reduced to hosts in the dominating sets. A dominating set is a subset of vertices of a graph such that every vertex of the graph is either a member of the subset or it must be a neighbor of at least one vertex in the subset \[57\]. To define mathematically, let us consider \( G = (V, E) \) be an undirected graph to represent mobile ad hoc network, where \( V \) represent set of mobile hosts (vertices) and \( E \) represent set of links between a pair of vertices. A subset \( D \) of a set of vertices \( V \) is called a dominating set of \( G \) if each node \( u \) in \( V \) (i.e. \( u \in V \)) is either in \( D \) (i.e. \( u \in D \)) or it is neighbor of some node \( v \) in \( D \) (i.e. \( \{u, v\} \in E \text{ and } v \in D \)). In the mobile network, members of dominating set are called gateway nodes and non-member nodes are called non-gateway nodes. A dominating set induces a subgraph in a graph. Finding a minimum connecting dominating set is an NP-complete problem for most of graphs. Some desirable features \[57\] for a dominating set are as follows:

- The algorithms for computing dominating sets should be distributed and simple.
- The computed dominating set should be connected and close to minimum.
- The dominating set should include all intermediate nodes of any shortest path.
4.2 Algorithm used to compute Dominating Sets

A number of algorithms [11, 12, 13, 14, 33, 37, 49, 57] by different authors have been proposed for finding the dominating sets. In this work we use the following localized and simplified algorithm [11]. The time complexity of this algorithm is $O(k)$ where $k$ is the maximum vertex degree in the network [11]. The computation complexity is of order $O(k^2)$ [11].

1. Initialize the gateway status of all nodes false in the beginning
2. For each node $X$ carry out the steps 3 to 12
3. Find the sub-graph of neighboring nodes of $X$ that has higher ID than ID of $X$. (The definition of ID is defined in the paragraph following the algorithm).
4. Apply the shortest path algorithm (Dijkstra’s algorithm) to check the connectivity of this sub-graph (obtained in step 3).
5. If the sub-graph is not connected
6. Set the gateway status of node $X$ true.
7. Else
8. Check whether each neighbor of $X$ is a neighbor of at least one node from the sub-graph
9. If the above condition in step 8 is true
10. Set the gateway status of node false.
11. Else
12. Set the gateway status of node true.

The node ID is defined [49] as follows:

$\text{ID} = (\text{degree}, x, y)$; where degree is the number of neighbors of a node and $x$ and $y$ are two coordinates representing the position of the node. A node with higher degree is considered as a higher ID node. In case two nodes have same degree, the node with higher $x$-coordinate is considered as higher ID node. In case $x$-coordinates of two nodes are also same, $y$-coordinates are tested to compare the ID of nodes.

The subgraph $G'$ induced by a dominating set have the following properties:

- The subgraph $G' = G[V']$ is a connected graph assuming that $G[V]$ is connected [57]. Where $V'$ is a subset of a set of vertices $V$ of a graph, that contains the gateway nodes only.
- The shortest path between any two nodes does not include any non gateway node as an intermediate node [57].
Example of Dominating Sets:

An example of dominating set computed through this algorithm for randomly selected twenty nodes (in the network area 640 x 480 units) is given below. Table 4.1 contains the information of nodes arranged in descending order of key (degree, X-coordinate, Y-coordinate). The column "Gateway Status" describes the status of a node in a network after running the algorithm. Only gateway nodes are members of dominating set. The status of non-gateway nodes is kept blank. The graph of these 20 nodes is shown in figure 4.1. The gateway nodes are also shown in dark black in figure 4.1 given below.

<table>
<thead>
<tr>
<th>Node No.</th>
<th>Degree</th>
<th>xCo</th>
<th>yCo</th>
<th>Gateway Status</th>
<th>Neighbor list</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>10</td>
<td>210</td>
<td>124</td>
<td>Gateway</td>
<td>1 2 3 4 5 7 10 13 16 17</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>254</td>
<td>192</td>
<td>Gateway</td>
<td>0 1 2 3 4 10 14 16 17</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>160</td>
<td>116</td>
<td>Gateway</td>
<td>1 2 5 7 10 13 14 16 17</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>299</td>
<td>55</td>
<td>Gateway</td>
<td>3 4 5 13 14 15 16 18</td>
</tr>
<tr>
<td>17</td>
<td>8</td>
<td>164</td>
<td>297</td>
<td>Gateway</td>
<td>0 1 2 3 5 7 11 14</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>96</td>
<td>170</td>
<td></td>
<td>2 3 5 7 13 14 16 17</td>
</tr>
<tr>
<td>16</td>
<td>7</td>
<td>133</td>
<td>69</td>
<td></td>
<td>1 3 5 7 10 13 14</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>97</td>
<td>264</td>
<td></td>
<td>0 1 3 5 7 14 17</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>38</td>
<td>181</td>
<td></td>
<td>1 2 3 13 14 16 17</td>
</tr>
<tr>
<td>0</td>
<td>6</td>
<td>253</td>
<td>334</td>
<td>Gateway</td>
<td>2 4 5 11 17 19</td>
</tr>
<tr>
<td>13</td>
<td>6</td>
<td>132</td>
<td>24</td>
<td></td>
<td>1 3 7 10 14 16</td>
</tr>
<tr>
<td>15</td>
<td>4</td>
<td>446</td>
<td>31</td>
<td>Gateway</td>
<td>9 10 12 18</td>
</tr>
<tr>
<td>18</td>
<td>4</td>
<td>443</td>
<td>31</td>
<td></td>
<td>9 10 12 15</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>364</td>
<td>228</td>
<td></td>
<td>0 5 10 14</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>587</td>
<td>77</td>
<td></td>
<td>9 15 18</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>571</td>
<td>75</td>
<td></td>
<td>12 15 18</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
<td>174</td>
<td>466</td>
<td></td>
<td>0 17 19</td>
</tr>
<tr>
<td>19</td>
<td>2</td>
<td>320</td>
<td>462</td>
<td></td>
<td>0 11</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>574</td>
<td>360</td>
<td>Gateway</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>503</td>
<td>406</td>
<td></td>
<td>6</td>
</tr>
</tbody>
</table>

Table 4.1: Node information in descending order of key value for dominating set
4.3 Dominating Set Based Routing

The efficiency of dominating set based routing mainly depends on the overheads involved in computing dominating sets and the size of dominating set. All routing approaches that use gateway host to route messages are referred to as dominating set based routing. The main advantage of dominating set based routing is that it simplifies the routing process to one in a smaller subnetwork generated from the dominating set. Only the gateway nodes participate in routing except the source and destination, since gateway nodes cover all nodes of the entire network. We have introduced the concept of dominating sets in position based routing algorithms namely VD-GEDIR, CH-MFR, R-DIR, DIR-16, DREAM, and LAR. The basic characteristics of the algorithms remain the same as described in [4. 31, 51]. Dominating sets impose an additional restriction in selecting the nodes that participate in routing from the request zone. The routing process in dominating set based routing generally involves the following steps:

Figure 4.1: Network with gateway nodes (dark black)
1. Identify the source and destination node
2. Determine the dominating set
3. Apply the actual algorithm (described in chapter 5) that uses only gateway nodes except source and destination nodes in forwarding and receiving message.

4.4 Mobility Management

The nodes in a mobile ad hoc network can move in any direction and with any speed. The mobility causes frequent changes in the topology of ad hoc networks. A particular node may switch off its communication utility to save its battery power and again switch it on after some time. The problem of nodes switching their transceiver on and off also leads to topological changes in the network. The topological changes in mobile ad hoc networks can be divided into three different categories namely - mobile node switch on, mobile node switch off, and mobile node movement (chapter 20 of [52]).

When a vertex should update or recalculate gateway status is the most challenging task in mobile environment? Another important task is how to update or recalculate the gateway status. The gateway update means that only individual mobile node update their gateway status. The gateway recalculation means that the entire network recalculates gateway status. When many mobile nodes in the network are in movement, the better choice is to recalculate gateway status of all nodes i.e. the dominating set is recalculated from scratch. On the other hand, if only a few mobile nodes are in movement gateway information is updated locally.

In our experiments we have recalculated the gateway status of all nodes from scratch in dominating set based routing before sending a data message. This is due to the reason that all nodes are moving at every simulation clock tick in a random direction. Therefore, there is a lot of movement and topology is changing frequently.
This chapter presents the design and implementation details of new proposed algorithms: VD-GEDIR, CH-MFR, DIR-16, R-DIR and implementation details of the existing algorithms DREAM and LAR. A brief survey of these algorithms is already given in chapter 2. In the beginning of the chapter, general concepts used in all algorithms are described. Rest of the chapter explains the algorithms one by one for both static and mobile network.

5.1 General Concepts

This section describes some important basic concepts and terms - routing, geocasting, expected zone, and request zone that are used in almost all algorithms. The understanding of these concepts and terms is necessary for analyzing/studying the algorithms in detail. Some mathematical concepts used in most of the algorithms are also explained in this section.

Routing is a process of sending a message from source to destination in hop-by-hop manner. Routing is responsible for producing routes that meet the service requirements of a network. Routing is a set of several components such as monitoring network topology, distributing location information for future route discovery, constructing and selecting routes, forwarding data packets to the selected routes etc. In position based routing knowing the new position of a destination due to mobility of nodes for sending messages is vital. To deal with the problem of mobility three routing approaches are proposed in the literature - reactive, proactive and hybrid. The location updates between neighboring nodes are common to all routing approaches so that every node always have recent information of its neighbors. The frequency of location updates depends on the mobility of nodes.
Chapter 5

• **Reactive**: In this approach a source node sends a short message for route discovery to the last known position of destination before transmitting the data message from source to destination. On receiving a route discovery message, destination responds by sending a route reply message to the source. After receiving the route reply message, source node starts sending data messages to destination. Intermediate nodes that participate in forwarding the route discovery and route reply messages update themselves with the position of source and destination respectively. Example of such an approach is LAR [31].

• **Proactive**: In this approach location update messages are floated in the network independent of routing. The source node transmits data message toward the last known position of destination and does not require a route discovery. Examples of such approach are DREAM [1, 4, 35].

• **Hybrid (Reactive + Proactive)**: In this approach location updates are also performed independent of routing. However, a source node sends a route discovery message to destination before transmitting the data messages. Here, a destination responds to the source by sending a route reply message on receiving the route discovery message. Then source sends the data message to destination [50].

**Geocasting** is a task of sending a message to all nodes located within a geographical region (e.g. circle or square). Geocasting can be defined as a variant of conventional multicasting where a message is sent to a group of hosts registered to a multicast group address. In geocasting, a message is sent to a group of nodes within a specified geographical region. The message is delivered to the geographical region and thereafter message is flooded inside the region. The message may be delivered to geographical region by using any of the routing approaches discussed above. In geocasting, destination is a geographical region instead of a single node.
Expected Zone is area in a network terrain where probability of finding a moving node is highest. Size of expected zone is calculated based on moving speed of a node and time elapsed since the last known position of destination. Suppose a source (S) wants to send a message to destination (D) at time \( t_1 \). Node S has the last position information of D updated at time \( t_0 \) (where \( t_1 > t_0 \)) and D is moving at an average speed \( V \). Now S can expect the new location of D in a region defined by \( V^*(t_1-t_0) \) centered at location of D at time \( t_0 \). This region is known as expected Zone since the probability of node D being in this region is very high. The expected zone can be a rectangle, a circle or any other shape.

Request Zone can be defined as an area drawn around the expected zone including source node. This area includes the nodes that are the best choices for any probable location of destination within the expected zone to forward a message. In routing/geocasting a source node forwards a message in a hop-by-hop manner to its neighbors inside the request zone to deliver a message to the final destination. The shape of a request zone may be a rectangle drawn from source at one end of diagonal and expected zone at other end of the diagonal. The shape may also be an angular one by drawing tangents from source on the circular expected zone.

Tangent angle: In most of the algorithms, a source node selects next hop(s) from its neighbors lying in the request zone. The request zone is an angular region between tangents drawn on the circular expected zone from the source. Therefore, the tangent angle is computed frequently to determine the neighbors lying in the request zone.

Tangent angle \( \alpha \) shown in figure 5.1 (given below) is calculated by the formula:

\[
\alpha = \sin^{-1} \left( \frac{r}{d} \right)
\]

where \( d = \sqrt{(S_x - D_x)^2 + (S_y - D_y)^2} \)

\( r = \) radius of circle.
**Angle between two lines:** In some algorithms a source needs to determine a neighbor that has closest direction towards destination. The direction is determined by calculating an angle between a straight line from source to destination and from source to neighbor. To calculate an angle between two lines let us consider a triangle ASD where the coordinates of A, S, and D are known as shown in figure 5.2 given below:

\[
\alpha = \cos^{-1}\left(\frac{SA^2 + SD^2 - AD^2}{2 \times SA \times SD}\right)
\]

where,
\[
SA = \sqrt{(S_x - A_x)^2 + (S_y - A_y)^2}
\]
\[
AD = \sqrt{(A_x - D_x)^2 + (A_y - D_y)^2}
\]
\[
SD = \sqrt{(S_x - D_x)^2 + (S_y - D_y)^2}
\]
Projected Distance: In some algorithms, a source needs to compute the projected distance of its neighbors to determine a neighbor with maximum projected distance towards destination. Let us consider figure 5.2 (above) to find the projected distance of point A on line SD. First draw a perpendicular from point A on line SD. Let the perpendicular intersects the line SD at point A'. The projected distance of point A on line SD is distance between S and A'. It can be calculated by the formula

\[ SA' = SA \times \cos(\alpha) \]

5.2 Implementation Details common to all Algorithms

In our implementation, we have created a unit graph of nodes. The direct connectivity information among nodes of the graph is computed and stored in a matrix called edge matrix. Calculating the Euclidian distance between every pair of nodes and comparing the distances with the transmission range compute this matrix. All the algorithms are implemented independently and are integrated with the basic module used for creating a network. The algorithms assume that a message is delivered between a source, destination pair if it finds connectivity between the source and destination.

In mobile network, pairs of source and destination are randomly selected such that distance between source and destination is more than the transmission range of a source. A circle (expected zone) is drawn based on the mobility speed and time elapsed since the last known location of destination. Further details are given in next chapter.

In static network the expected zone is realized by drawing a circle around destination since nodes are fixed and mobility speed is zero. This circle is also used as geographic region for geocasting. Size of a circle varies by changing the percentage of nodes (10%, 20%, and 30%) inside the circle. A pair of source and destination is selected
randomly such that the distance between source and destination node is more than the sum of transmission range plus radius of a destination circle. This ensures that none of the probable position of destination in the circle is within direct transmission range of the source. Further, this also ensures the fundamental objective of the routing that there may be multiple hops between source and destination.

In a multi hop routing a message delivery is considered to be successful in case one of the following conditions is satisfied:

- A message is delivered to a node that is inside the circle and is connected with destination.
- A message is delivered to a node that is outside the circle and its transmission range covers entire circle and actual destination.

Connectivity of the destination with a node that is lying inside the circle is determined using shortest path algorithm.

In case of multiple routes between a source and destination (in static or mobile network), a node is never marked more than once for forwarding the data. This ensures that the marked nodes between a source and a destination are distinct. Marking of distinct nodes also ensures to avoid loop formation. A marked node does not further mark its neighbors if any one of the three conditions A, B, and C is satisfied.

A. A marked node is isolated i.e. it has no neighbors to forward the data.
B. A marked node is inside the circle.
C. The transmission range of a marked node covers the entire circle.

When the entire circle falls within the transmission range, a marked node can transmit data message directly to all possible destination locations inside the destination circle. While testing programs in the initial stages of implementation, cases were observed where the nodes lying beyond the circle were also marked. This caused failure of a
message delivery. One such example is graphically shown in figure 5.1 given below in which the transmission range of marked node No. 92 covers the entire destination circle.

Figure 5.3: Expected zone entirely within transmission range

5.3 Algorithmic Details for Static and Mobile Networks

This section describes the algorithms in general and their implementation in static and mobile networks.

5.3.1 Voronoi Diagram – Geographic Distance Routing (VD-GEDIR)

This algorithm is based on geographic distance routing (GEDIR) described in paper [47]. In GEDIR, at each hop a source node forwards a message to its neighbor that is closest to the destination. The closest neighbor is the one that has least geographic distance from the destination. This is determined by comparing the geographic distances between destination and neighbors of the source.
VD-GEDIR algorithm [51] modifies the definition of request zone for GEDIR. In this algorithm a source forwards a message to its neighbors that are closest to a possible position of the destination in expected zone. These neighbors for VD-GEDIR algorithm are determined by using the concept of Voronoi diagram for set of nodes of network.

Authors in [51] have also suggested an alternative approach for implementing VD-GEDIR algorithm. In the alternative approach, certain number of points on the circle boundary is selected as probable destinations by making an approximate decision. Then a closest next hop neighbor for each probable destination is determined using GEDIR as basic algorithm for forwarding the message. However, some of the desired neighbors may not be discovered, and computation time might increase.

We have implemented plane intersect method of constructing Voronoi diagram to realize Voronoi diagram algorithm for routing and geocasting. Voronoi diagram is also implemented by approximating the voronoi regions. In approximation method we have taken 16 equally spaced points on the boundary of the circle assuming that destination may be anywhere inside the circle including these 16 positions. We refer to the Voronoi diagram algorithm as VD and the approximation algorithm as VD-GEDIR.

Authors of [51] have suggested some modification in VD-GEDIR to make it loop free when past traffic is not memorized.

5.3.1.1 VD Algorithm (Original) for Static Network

1. Select a source and destination randomly.
2. Draw a circle around destination.
3. Initialize tmp-source with source.
4. If tmp-source is isolated or covers entire circle then go to step 9.
5. If tmp-source is inside circle then check destination connectivity and go to 9.
7. Mark selected nodes as next hop that are distinct.
8. Insert marked node in the queue LMN.
9. If not EMPTY(LMN) then (tmp-source ← next marked node) and go to step 4.
10. Stop.
5.3.1.2 VD-GEDIR Algorithm (Approximation) for Static Network

1. Select a source and destination randomly.
2. Draw a circle around destination.
3. Compute 16 equally spaced points as probable destination on circle boundary.
4. Initialize tmp-source with source.
5. If tmp-source is isolated or covers entire circle then go to step 12.
6. If tmp-source is inside circle then check destination connectivity and go to 12.
7. Select a probable destination.
8. For each neighbor of tmp-source, calculate distance between a neighbor and selected probable destination.
9. Mark a neighbor node that has minimum distance and is not already marked as next hop.
10. Insert marked node in the queue LMN.
11. Repeat steps 7-10 for all 16 probable destinations.
12. If not EMPTY(LMN) then (tmp-source ← next marked node) and go to step 5.
13. Stop.

5.3.1.3 VD Algorithm (Original) for Mobile Network

1. Read an event notice from future event list (FEL) (described in next chapter).
2. If event type is “Node Movement” execute routine for movement of nodes.
3. If event type is “Location updates” execute routine for location updates.
4. If event type is “Data transmission” go to step 6.
5. If not EOF (FEL) go to step 1, otherwise go to step 18.
6. Select a source and destination.
7. Initialize tmp-source with source.
8. If (tmp-source = destination) then go to step 17.
10. If (timeout > a threshold value) then execute recovery and go to step 5.
11. Draw a circle around destination.
12. If tmp-source is isolated or covers entire circle then go to step 17.
13. If tmp-source is inside circle then check destination connectivity and go to 17.
15. Mark selected nodes as next hop that are distinct.
16. Insert marked node in queue (LMN).
17. If not EMPTY(LMN) then (tmp-source ← next marked node) and go to step 8.
18. Stop.

5.3.1.4 VD-GEDIR Algorithm (Approximation) for Mobile Network

1. Read an event notice from future event list (FEL) (described in next chapter).
2. If event type is “Node Movement” execute routine for movement of nodes.
3. If event type is “Location updates” execute routine for location updates.
4. If event type is “Data transmission” go to step 6
5. If not EOF (FEL) go to step 1, otherwise go to step 21.
6. Select a source and destination.
7. Initialize tmp-source with source
8. if (tmp-source = destination) then go to step 20
9. Calculate value of timeOut
10. If (timeout > a threshold value) then execute recovery and go to step 5
11. Draw a circle around destination
12. Compute 16 equally spaced points as probable destination on circle boundary
13. If tmp-source is isolated or covers entire circle then go to step 20
14. If tmp-source is inside circle then check destination connectivity and go to 20
15. Select a probable destination
16. For each neighbor of tmp-source, calculate distance between a neighbor and selected probable destination
17. Mark a neighbor node as next hop that has minimum distance and is distinct
18. Insert marked node in queue (LMN)
19. Repeat steps 15-18 for all 16 probable destinations
20. If not EMPTY(LMN) then (tmp-source ~ next marked node) and go to step 8 otherwise go to step 5
21. Stop

5.3.2 Convex Hull – Most Forward Progress within Radius (CH-MFR)

This algorithm is based on most forward within radius routing algorithm (MFR), which is described in paper [47]. In MFR the message is forwarded to the neighbor with greatest progress. The progress is defined as the distance between the source and neighbor node projected onto a line drawn from source toward the destination.

The CH-MFR algorithm [48, 51] modifies the definition of request zone for MFR. In this algorithm, a source forwards a message to its neighbors that have most forward progress for a probable destination. These neighbors for MFR algorithm are determined by using the concept of convex hull for the set of nodes of network. Convex hull of a set of points is a smallest convex polygon for which each point in the set is either on the boundary of the convex polygon or in its interior.

Authors in [51] have also suggested an alternative approach for implementing CH-MFR algorithm. In the alternative approach, certain number of points on the circle
boundary is selected as probable destinations by making an approximate decision. Then a closest next hop neighbor for each probable destination is determined using MFR as basic algorithm for forwarding the message. However, some of the desired neighbors may not be discovered, and computation time might increase.

We have implemented Jarvis's march method of constructing convex hull to realize convex hull algorithm for routing and geocasting. Convex Hull algorithm is also implemented by approximation method. In Approximation method we have taken 16 equally spaced points on the boundary of the circle assuming that destination may be anywhere inside the circle including these 16 positions. The Convex Hull algorithm implemented in original is referred to as CH while approximation method is referred to as CH-MFR.

Paper [51] proposes that an additional restriction of dot product can be used in CH and CH-MFR to select a neighbor node closer to destination. Dot product restriction states that suppose a node A receive a message from neighboring node C. A will forward the message to its neighbors B only when B satisfy the condition \(DBDA < DADe\) where D is destination. The loop-free property of CH-MFR algorithm follows from the proof of loop-free property of MFR given in [47].

5.3.2.1 CH Algorithm (Original) for Static Network

1. Select a source and destination randomly.
2. Draw a circle around destination.
3. Initialize tmp-source with source
4. if tmp-source is isolated or covers entire circle then go to step 9
5. If tmp-source is inside circle then check destination connectivity and go to 9
6. Select neighbors of tmp-source by constructing CH using Jarvis's march method
7. Mark selected nodes as next hop that satisfy dot product condition, and are unique
8. Insert marked node in the queue (LMN)
9. If not EMPTY(LMN) then (tmp-source ← next marked node) and go to step 5
10. Stop
5.3.2.2 CH-MFR Algorithm (Approximation) for Static Network

1. Select a source and destination randomly.
2. Draw a circle around destination.
3. Compute 16 equally spaced points as probable destination on circle boundary.
4. Initialize tmp-source with source
5. if tmp-source is isolated or covers entire circle then go to step 12
6. If tmp-source is inside circle then check destination connectivity and go to 12
7. Select a probable destination.
8. For each neighbor of tmp-source, calculate projected distance of a neighbor by drawing a perpendicular on line joining tmp-source and expected destination
9. Mark a neighbor as next hop that has maximum forward projected distance, and satisfy dot product condition, and is distinct
10. Insert marked node in queue (LMN)
11. Repeat steps 7-10 for all 16 probable destinations
12. If not EMPTY(LMN) then (tmp-source ← next marked node) and go to step 5
13. Stop

5.3.2.3 CH Algorithm (Original) for Mobile Network

1. Read an event notice from future event list (FEL)
2. If event type is "Node Movement" execute routine for movement of nodes
3. If event type is "Location updates" execute routine for location updates
4. If event type is "Data transmission" go to step 6
5. If not EOF (FEL) go to step 1, otherwise go to step 18.
6. Select a source and destination.
7. Initialize tmp-source with source
8. if (tmp-source = destination) then go to step 17
9. Calculate value of timeOut
10. If (timeOut > a threshold value) then execute recovery and go to step 5
11. Draw a circle around destination
12. if tmp-source is isolated or covers entire circle then go to step 17
13. If tmp-source is inside circle then check destination connectivity and go to 17
14. Select neighbors of tmp-source by constructing CH using Jarvis's march method
15. Mark selected nodes as next hop that satisfy dot product condition, and are unique
16. Insert marked node in queue (LMN)
17. If not EMPTY(LMN) then (tmp-source ← next marked node) and go to step 8 otherwise go to step 5
18. Stop
5.3.2.4 CH-MFR Algorithm (Approximation) for Mobile Network

1. Read an event notice from future event list (FEL)
2. If event type is "Node Movement" execute routine for movement of nodes
3. If event type is "Location updates" execute routine for location updates
4. If event type is "Data transmission" go to step 6
5. If not EOF (FEL) go to step 1, otherwise go to step 21.
6. Select a source and destination.
7. Initialize tmp-source with source
8. if (tmp-source = destination) then go to step 20
9. Calculate value of timeOut
10. If (timeOut > a threshold value) then execute recovery and go to step 5
11. Draw a circle around destination
12. Compute 16 equally spaced points as probable destination on circle boundary
13. if tmp-source is isolated or covers entire circle then go to step 20
14. If tmp-source is inside circle then check destination connectivity and go to 20
15. Select a probable destination
16. For each neighbor of tmp-source, calculate projected distance of a neighbor by drawing a perpendicular on line joining tmp-source and probable destination
17. Mark a neighbor node as next hop that has maximum forward projected distance, satisfy dot product condition, and is distinct
18. Insert marked node in queue (LMN)
19. Repeat steps 15-18 for all 16 probable destinations
20. If not EMPTY(LMN) then (tmp-source <- next marked node) and go to step 8 otherwise go to step 5
21. Stop

5.3.3 Range Directional (R-DIR)

The R-DIR algorithm [51] modifies the definition of request zone for the direction based routing algorithms. In this algorithm a source forwards a message to its neighbors that have closest direction for a probable destination.

In R-DIR [48, 51], a source node makes an angular zone by drawing tangents from the source on both sides of the circle (expected zone). The request zone consists of the nodes lying inside the angular zone and at the most two nodes that have closer direction towards the possible destinations but outside the angular zone (see figure 2.5 in chapter 2). The process of marking of neighbor nodes from outside the tangents is more clearly specified as follows:
A. First, the best node i.e. the node that makes maximum angle with a tangent (lying inside the angular zone) is determined. The angle between tangent and the best node is called alpha.

B. Then among all the neighbor nodes that are outside the tangent, the node that makes the minimum angle with the tangent is determined. This minimum angle is called beta.

C. If the angle found in step B (beta) is less then the angle found in step A (alpha) and angle [beta + tangent angle] is less than the 90 degree then node determined in step B is marked.

D. Steps A, B, and C are applicable for both tangents if two tangents are not same as in case when radius of circle is not zero.

E. There are cases when angular zone is empty. In that case the minimum angle found in step B [beta + tangent angle] is only compared with 90 degree. If the minimum angle found in step B is less then the tangent angle then that node is marked.

F. If radius of circle is equal to zero then two tangents overlap each other and in fact there is only one line joining the source and destination. In this situation only one neighbor node will be marked which will makes smallest angle with line joining the source and destination.

Paper [51] proposes that an additional restriction of dot product can also be used in R-DIR to select a neighbor node closer to destination. Dot product restriction states that suppose a node A receive a message from neighboring node C. A will forward the message to its neighbors B only when B satisfy the condition $\angle DBDA < \angle DA' DC$ where D is destination.

5.3.3.1 R-DIR Algorithm for Static Network

1. Select a source and destination randomly.
2. Draw a circle around destination.
3. Initialize tmp-source with source
4. if tmp-source is isolated or covers entire circle then go to step 11
5. If tmp-source is inside circle then check destination connectivity and go to 11
5.3.3.2 R-DIR Algorithm for Mobile Network

1. Read an event notice from future event list (FEL)
2. If event type is "Node Movement" execute routine for movement of nodes
3. If event type is "Location updates" execute routine for location updates
4. If event type is "Data transmission" go to step 6
5. If not EOF (FEL) go to step 1, otherwise go to step 20
6. Select a source and destination.
7. Initialize tmp-source with source
8. if (tmp-source = destination) then go to step 19
9. Calculate value of timeOut
10. If (timeOut > a threshold value) then execute recovery and go to step 5
11. Draw a circle around destination.
12. if tmp-source is isolated or covers entire circle then go to step 19
13. If tmp-source is inside circle then check destination connectivity and go to 19
14. Draw tangents on both sides of the circle from tmp-source
15. Select all neighbor nodes lying inside two tangents containing circle
16. Select suitable neighbor(s) outside the tangents that fulfill the stated criteria and satisfy dot product condition.
17. Mark the selected nodes (in steps 15-16) as next hops that are distinct
18. Insert marked nodes in queue (LMN)
19. If not EMPTY(LMN) then (tmp-source ← next marked node) and go to step 8
   otherwise go to step 5
20. Stop

5.3.4 DIR-16

E. Kranakis, H. Singh, and J. Urrutia in [32] proposed a compass routing method to deliver a message from source to destination in a geometric network. Later on, Ivan Stojmenovic called this method as DIR in his paper [47], and compared the performance of DIR with GEDIR, MFR, LAR, and SP.
In DIR, a source S to send message to destination D, first calculates the direction of all its neighbors based on their locations with reference to the location of D. Then S forwards the message to its neighbor that is closest to the direction D among all neighbors. The intermediate nodes repeat the same process until the message is delivered to D (if accessible).

Here we propose some modification in this algorithm to make it more suitable for routing and geocasting by incorporating the concept of expected zone and request zone. In the modified approach we select 16 equally spaced points on the boundary of an expected zone as probable destinations and further assume that destination may be anywhere inside the circle including these 16 positions. Next hop neighbors for each probable destination is selected using DIR as basic algorithms for forwarding the message. So no explicit request zone is drawn for this case.

Paper [47] discusses that DIR may form temporary loops in some cases when operated in memory less mode. Similarly the probability of temporary loop formation cannot be eliminated in DIR-16 in memory less mode since DIR-16 is based on DIR method. Therefore we have implemented DIR with memorization of past traffic to avoid loop formation.

5.3.4.1 DIR-16 Algorithm for Static Network

1. Select a source and destination randomly.
2. Draw a circle around destination.
3. Compute 16 equally spaced points as probable destination on circle boundary.
4. Initialize tmp-source with source
5. If tmp-source is isolated or covers entire circle then go to step 12
6. If tmp-source is inside circle then check destination connectivity and go to 12
7. Select a probable destination.
8. For each neighbor of tmp-source, calculate angle between line joining tmp-source and a neighbor node, and line joining tmp-source and probable destination
9. Mark a neighbor as next hop that has minimum angle and is distinct
10. Insert marked node in queue (LMN)
11. Repeat steps 7-10 for all 16 probable destinations
12. If not EMPTY(LMN) then (tmp-source ← next marked node) and go to step 5
13. Stop
5.3.4.2 DIR-16 Algorithm for Mobile Network

1. Read an event notice from future event list (FEL)
2. If event type is “Node Movement” execute routine for movement of nodes
3. If event type is “Location updates” execute routine for location updates
4. If event type is “Data transmission” go to step 6
5. If not EOF (FEL) go to step 1, otherwise go to step 21.
6. Select a source and destination.
7. Initialize tmp-source with source
8. if (tmp-source = destination) then go to step 20
9. Calculate value of timeOut
10. If (timeOut > a threshold value) then execute recovery and go to step 5
11. Draw a circle around destination
12. Compute 16 equally spaced points as probable destination on circle boundary
13. if tmp-source is isolated or covers entire circle then go to step 20
14. If tmp-source is inside circle then check destination connectivity and go to 20
15. Select a probable destination
16. For each neighbor of tmp-source, calculate angle between line joining tmp-source and a neighbor node, and line joining tmp-source and probable destination
17. Mark a neighbor as next hop that has minimum angle and is distinct
18. Insert marked node in queue (LMN)
19. Repeat steps 15-18 for all 16 probable destinations
20. If not EMPTY(LMN) then (tmp-source ← next marked node) and go to step 8
   otherwise go to step 5
21. Stop

5.3.5 Location Aided Routing (LAR)

The authors of Location Aided Routing (LAR) [44] have presented two schemes for transmitting a message from a source to destination. We refer the first scheme as LAR1 and second scheme as LAR2. These schemes differ in formation of their request zone and in the process of marking nodes to forward message within the request zone.

Scheme 1 (LAR1)

In this scheme a circular expected zone is formed around the known location of the destination. Further, the source defines a rectangular request zone by drawing a smallest rectangle that includes the location of the initial source and expected zone.
The source node forwards the message to all its neighbors within the request zone. This process is repeated until the message is delivered to destination. The intermediate selected/ marked nodes use same expected zone and request zone created in the beginning i.e. these nodes do not redraw expected zone and request zone. In case the destination node is not found within the request zone, the message is flooded in the entire network.

5.3.5.1 LARI Algorithm for Static Network

1. Select a source and destination randomly.
2. Initialize tmp-source with source.
3. Draw a circle around destination.
4. Draw a smallest rectangle including the tmp-source position and circle.
5. if tmp-source is isolated or covers entire circle then go to step 10.
6. If tmp-source is inside circle then check destination connectivity and go to 10.
7. Select all neighbor nodes lying inside the rectangle (drawn in step 4).
8. Mark the selected nodes (in steps 7) as next hops that are distinct.
9. Insert marked nodes in queue (LMN).
10. If not EMPTY(LMN) then (tmp-source ← next marked node) and go to step 5.
11. Stop.

5.3.5.2 LARI Algorithm for Mobile Network

1. Read an event notice from future event list (FEL).
2. If event type is "Node Movement" execute routine for movement of nodes.
3. If event type is "Location updates" execute routine for location updates.
4. If event type is "Data transmission" go to step 6.
5. If not EOF (FEL) go to step 1, otherwise go to step 19.
6. Select a source and destination.
7. Initialize tmp-source with source.
8. Draw a circle around destination.
9. Draw a smallest rectangle including the tmp-source position and circle.
10. if (tmp-source = destination) then go to step 18.
11. Calculate value of timeOut.
12. If (timeOut > a threshold value) then execute recovery and go to step 5.
13. if tmp-source is isolated or covers entire circle then go to step 18.
14. If tmp-source is inside circle then check destination connectivity and go to 18.
15. Select all neighbor nodes lying inside the rectangle (drawn in step 9).
16. Mark the selected nodes (in steps 15) as next hops that are distinct.
17. Insert marked nodes in queue (LMN).
18. If not EMPTY(LMN) then (tmp-source ← next marked node) and go to step 5.
19. Stop.

Algorithm Details 59
Scheme 2 (LAR2)

In this scheme a source does not define a specific expected zone and explicit request zone as defined in LAR1. The source forwards a message to its neighbors that are closer to destination than the source. This process is repeated until the destination is reached. On failure of delivery flooding is used to transmit the message to the destination.

We have drawn an expected zone around the destination in the same way as in LAR1 for experimentation. Although, there is no concept of expected zone in original algorithm LAR2, however it is drawn to count the number of marked nodes outside the circle for comparing the results with other algorithms. This does not impinge on the original logic of LAR2.

5.3.5.3 LAR2 Algorithm for Static Network

1. Select source and destination randomly.
2. Initialize tmp-source with source
3. Draw a circle around destination.
4. Calculate distance \( X_s \) from tmp-source to destination
5. If (Transmission range \( \geq X_s \)) then go to step 10
6. Calculate distance \( X_i \) between a neighbor of tmp-source and destination
7. If \( (X_s > X_i) \) then mark the neighbor node as next hop if it is distinct
8. Insert marked node in queue (LMN)
9. Repeat steps 7-9 for all neighbors of tmp-source
10. If not EMPTY(LMN) then (tmp-source \( \leftarrow \) next marked node) and go to step 4
11. Stop

5.3.5.4 LAR2 Algorithm for Mobile Network

1. Read an event notice from future event list (FEL)
2. If event type is “Node Movement” execute routine for movement of nodes
3. If event type is “Location updates” execute routine for location updates
4. If event type is “Data transmission” go to step 6
5. If not EOF (FEL) go to step 1, otherwise go to step 19
6. Select a source and destination.
7. Initialize tmp-source with source
8. Draw a circle around destination.
9. if (tmp-source = destination) then go to step 18
10. Calculate value of timeout
11. If (timeout > a threshold value) then execute recovery and go to step 5
12. Calculate distance $X_s$ from tmp-source to destination
13. If (Transmission range $\geq X_s$) then go to step 18
14. Calculate distance $X_i$ between a neighbor of tmp-source and destination
15. If ($X_s > X_i$) then mark the neighbor node as next hop if it is distinct
16. Insert marked node in queue (LMN)
17. Repeat steps 7-9 for all neighbors of tmp-source
18. If not EMPTY(LMN) then (tmp-source $\leftarrow$ next marked node) and go to step 9
   otherwise go to step 5
19. Stop

5.3.6 Distance Routing Effect Algorithms for Mobility (DREAM)

In this algorithm [22] the definition of expected zone remains the same as described in LAR. However, the request zone is an angular area instead of a rectangular area. The tangents are drawn from source on both sides of the circle to form the angular area. A source node transmits a message to all its neighbors. But only the neighbors that are inside the request zone transmit the message further. Here, each neighbor node computes the request zone based on the information of destination from their own location tables before forwarding the message to its neighbors. This procedure is repeated until the message is delivered to the destination. If destination is not found, algorithm starts recovery method using flooding.

5.3.6.1 DREAM Algorithm for Static Network

1. Select a source and destination randomly.
2. Draw a circle around destination.
3. Initialize tmp-source with source
4. if tmp-source is isolated or covers entire circle then go to step 10
5. If tmp-source is inside circle then check destination connectivity and go to 10
6. Draw tangents on both sides of the circle from tmp-source
7. Select all neighbor nodes of lying inside two tangents containing circle
8. Mark the selected nodes (in steps 7) as next hops that are distinct
9. Insert marked nodes in queue (LMN)
10. If not EMPTY(LMN) then (tmp-source $\leftarrow$ next marked node) and go to step 4
11. Stop

Algorithm Details 61
5.3.6.2 DREAM Algorithm for Mobile Network

1. Read an event notice from future event list (FEL)
2. If event type is "Node Movement" execute routine for movement of nodes
3. If event type is "Location updates" execute routine for location updates
4. If event type is "Data transmission" go to step 6
5. If not EOF (FEL) go to step 1, otherwise go to step 19
6. Select a source and destination.
7. Initialize tmp-source with source
8. if (tmp-source = destination) then go to step 18
9. Calculate value of timeout
10. If (timeout > a threshold value) then execute recovery and go to step 5
11. Draw a circle around destination.
12. if tmp-source is isolated or covers entire circle then go to step 18
13. If tmp-source is inside circle then check destination connectivity and go to 18
14. Draw tangents on both sides of the circle from tmp-source
15. Select all neighbor nodes lying inside two tangents containing circle
16. Mark the selected nodes (in steps 15) as next hops that are distinct
17. Insert marked nodes in queue (LMN)
18. If not EMPTY(LMN) then (tmp-source ← next marked node) and go to step 8 otherwise go to step 5
19. Stop