

# An Energy Efficient Routing Approach for Wireless Sensor Networks

Bahlul Haider, Kokichi Sugihara

Department of Mathematical Informatics,  
The University of Tokyo  
bahlul\_haider@mist.i.u-tokyo.ac.jp,  
sugihara@mist.i.u-tokyo.ac.jp

**Abstract.** Energy efficiency is the most important aspect in wireless sensor networks. Energy consumption could be reduced by an efficient routing topology of the sensor network. Further energy consumption could be reduced by taking localized routing decision. In this paper we propose an energy efficient routing for wireless sensor networks called almost Delaunay triangulation routing approach, which meets both the energy efficiency and localized routing criteria. The architecture of the routing tree will be dynamic even though the underlying graph is static and calculated only once. Experimental results show that the performance of the proposed approach is good for uniform random distribution of the sensor nodes.

## 1. Introduction

A sensor network is a network of many smart devices, called sensor nodes, which are distributed in order to perform an application-oriented global task. The primary component of a sensor network is the sensor. Sensor nodes are used to monitor some real world physical phenomenon like pressure, temperature, humidity, presence/absence of something, vibration, intensity, sound, pressure, motion and pollutant etc. Each of the sensor nodes is small and inexpensive but smart enough to perform several tasks. The smart tiny sensor nodes are equipped with microcontroller, a radio transmitter and an energy source. The most important design and implementation requirements of a typical sensor network are energy efficiency, routing decision type, memory capacity, computational speed and bandwidth. Sometimes a special central node is deployed to control all the operations of the network.

Wireless sensor network, widely known as WSN, has become a wide area of research nowadays as the processing and storage technology became mature and feasible recently. We can now think of deploying hundreds of thousands of cheap sensor nodes to a target area to sense some special types of information. Sensor networks are used in a wide range of areas such as sensing ocean floor activities, volcanic activities, and phenomena in Mars etc. In most of the cases, sensor nodes are deployed in places where the physical condition is very adverse and human cannot go there. For this reason, there is no pre-existing network topology available in the target area. The major problem of the sensor network is the energy consumption. The battery technol-

ogy is much lagging behind than the processing and storage technology. For this reason, the primary research on sensor network is how to efficiently use the power consumption of each of the sensor nodes to maximize the total lifetime of the network. In most of the cases, it is not feasible or even impossible to recharge the battery of the sensor nodes. The power consumption of the sensor nodes mostly depends on the routing of information in the network. Other means of power consumption include sensing and processing of the information. There is not much scope to reduce the power consumption related to sensing and processing. So, most of the research about sensor networks mainly concentrates on the power consumption due to the routing of information and there is still much scope to improve. Compression could reduce the amount of data transmitted in sensor network. However, data compression is out of the scope of this paper and hence we will not discuss about data compression.

In this paper, we propose an algorithm based on Delaunay triangulation to organize the topology of the sensor network. We name it as almost Delaunay triangulation. The almost Delaunay triangulation routing graph is generated locally and the graph is generated only once. At each round, the sensor nodes use a subgraph of the almost Delaunay triangulation which is very easy to generate and also very much cost effective. The proposed algorithm shows good performance when the distribution of the sensor nodes is uniform. This paper is based on [13] with some enhancements.

## 2. Background

The background of sensor network research is relatively new, as the processing and storage technology has matured recently to make the numerous sensor nodes feasible. Research on sensor network includes preliminary ideas about how the sensor network should function [2][4][9][23][25] and also how to efficiently use the energy [6][8][19][25][26][28]. References [17] and [20] surveyed the coverage problem in wireless network. At first sight one may think that the architecture used in existing ad-hoc networks could be reused in sensor network. However, ad-hoc networks and sensor networks are completely different in many ways. Firstly, in sensor network sensor nodes do not have any unique identification unlike ad-hoc network. Secondly, in ad-hoc network the mobile nodes could be charged if needed (like we charge our cell phones every day). Power is not a problem in ad-hoc network. On the other hand, in sensor network it is not feasible to recharge the batteries of the sensor nodes. We could use data abstraction in sensor network, as the neighboring sensor nodes senses almost the same data [25] and neighboring data is correlated. These reasons make sensor network different from the ad-hoc network and give some new challenges to the researchers. A lot of recent work is done on energy efficient routings [2][8][11][19][21][25][26]. Some research divides the network in several rounds [21][25] and in each round the topology is changed. Reference [21] uses a greedy approach to calculate a near optimal cycle among the sensor nodes. Some research also suggested information gathering models to reduce the amount of information transferred. In our almost Delaunay triangulation routing we also divide the sensing time into rounds. In each round we only change the head, which is the receiver of all

information in the network. The head of each round communicates with the base station. Base station is usually very far from the target area. In our proposed approach, we do not have to calculate the network topology in every round.

### 3. Preliminaries

#### 3.1. Assumptions

In the sensor network we assume the following properties.

1. The minimum number of sensor needed to cover an area is given by the following equation

$$N_{\min} = \frac{A \times \sqrt{27}}{2 \times r^2 \times \pi^2}$$

where  $A$  is the target area,  $N_{\min}$  is the minimum number of sensor nodes, and  $r$  is the sensing range of each node [18]. We assume that in our target area we have  $N$  nodes, where  $N > N_{\min}$ .

2. The sensor nodes sense information continuously and send the information continuously to the next node towards the head.

#### 3.2. Our Goal

Our goal in sensor network includes the following.

1. Minimize the summation of the squared distance of the edges in the routing tree at each round, which implies the maximization of the lifetime of the network, because the power consumption is proportional to the squared distance [22].
2. Minimize the diameter of the tree in each round (long paths in the graph means it will take long time to reach the information to the head).
3. Find the tree by local computation only. This is because, neither it is feasible to use a central supervising node in sensor network, nor every node has complete information about the network.

#### 3.3. Power sink

The power sink in sensor network includes sensing, processing and transceiving of information within the network. Most of the power is consumed in transmission. The energy consumption to transfer a unit information to a distance  $d$  is proportional to  $d^n$ , where  $2 \leq n \leq 4$  [22]. So, we should take small  $d$ 's whenever possible.

#### 3.4. Single vs. Multi Hop

In single hop sensor network, each node directly sends information to the central node or the head. References [25] and [28] analyzed single hop sensor network. In

single hop sensor network, the node that is far from the head consumes more power. On the other hand in the multi hop sensor network several intermediate nodes are used to route the information to the head. Reference [21] used multi hop sensor network. In multi hop sensor network, the node that is nearest to the head consumes more power as it has to handle a lots of child nodes. In the case of multi hop, it will take more time to reach the information to the head, because each hop requires some additional time. So, we need some trade off between single hop and multi hop. We should use multi hop when possible to reduce power consumption. We should also keep in mind not to make too much long paths to reduce information propagation delay. If we allow multi hop in the graph, minimum spanning tree (MST) is the best solution. We are not considering about the MST, because it contains very long path compared to the number of nodes in the graph. If we could restrict the diameter in the MST, then it would be our desired solution. However, Bounded Diameter Minimum Spanning Tree (BDMST) is NP-hard [12]. So, no efficient algorithm is likely to exist for BDMST.

### 3.5. Central vs. Distributed Approach

In central approach, the routing graph and the head is selected by one or few of the sensor nodes. On the other hand, in distributed approach the graph and the head is selected by all the nodes in the sensor network. Central approach is not feasible in sensor network, as the central node will become the bottleneck of the whole network. The nodes should be able to organize themselves in a distributed and localized fashion and the loss of some of the nodes will not have much effect on the remaining network. The whole network should work well even if some nodes fail or expire.

### 3.6. Static vs. Dynamic Topology

Some sensor network algorithm changes their networks dynamically and some remains static for the whole lifetime of the network. If we use static topology, then the energy consumption will not be evenly distributed among all the nodes of the network. Some nodes will consume more energy than the others and will die much earlier. To avoid situations like this, we will use dynamic topology. Thus, the load of the network will be evenly distributed among all the nodes.

### 3.7. Sink based vs. Ad-hoc

Sink based sensor networks use one or more sink nodes which are also known as heads. These sinks gather all the information and manage the whole network. The sink nodes are usually the receivers of all node's information in particular region. On the other hand, there is no sink in ad-hoc networks. All sensor nodes are equal. Our proposed algorithm is sink-based, but we change the sink (head) in each round.

### 3.8. Delaunay Triangulation

A Delaunay triangulation for a set  $P$  of points in the plane is a triangulation  $DT(P)$  such that no points in  $P$  is inside the circumcircle of any triangle in  $DT(P)$ . Delaunay triangulation has some interesting properties that make it unique. Firstly, external Delaunay edges in  $DT(P)$  constitute the boundary of the convex hull of  $P$ . Secondly, all circumcircles of Delaunay triangulation are empty. Moreover, some important graphs like Gabriel graph, the relative neighbor graph, Euclidean minimum spanning tree and the nearest neighbor graph are subgraphs of Delaunay triangulation [1]. Delaunay triangulation maximizes the minimum angle of all the angles of the triangles in the triangulation. In Delaunay triangulation the expected number of node degree is less than or equal to 6 for all network sizes [1]. Addition and deletion of nodes has only local effects. Another interesting property of Delaunay triangulation is that compass routing is possible in Delaunay triangulation [3]. Figure 1 shows the Delaunay triangulation. Dual graph of the Delaunay triangulation is the Voronoi diagram. Voronoi diagram is shown in shaded line in Figure 1. Each polygon in the Voronoi graph is governed by a vertex.

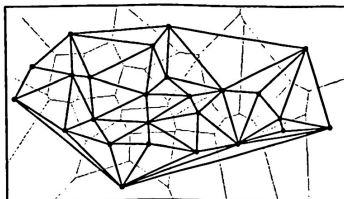


Figure 1. Delaunay triangulation.

### 3.9. $E_{MSTM\text{MAX}}$

Suppose that there are  $N$  points in an area  $A$ . Let  $E_{MSTM\text{MAX}}$  represents the longest edge in the minimum spanning tree (MST). Approximating the value of  $E_{MSTM\text{MAX}}$  is important, because the connectivity of the routing tree in our proposed approach depends on the good approximation of  $E_{MSTM\text{MAX}}$ .

### 3.10. Compass Routing

The *compass routing* (CR) is a local routing strategy which always moves the packet to the vertex that minimizes the angle over all vertices adjacent to the current node towards the destination node. Delaunay triangulation always guarantees that the packet will reach to the destination [24]. Compass routing is shown in figure 2.

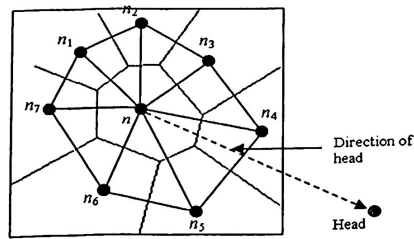


Figure 2. Compass routing.

The current node  $n$  has seven neighbors from  $n_1$  to  $n_7$ . Node  $n$  knows about the location of the head. So, it can easily calculate the direction of the head. In Figure 2, the node  $n_4$  is in the closest direction of the head. So, node  $n$  will send its sensed information to  $n_4$ . Here the question arises: will compass routing guarantee that we will finally reach to the destination? The answer is “yes” for Delaunay triangulation, but is not always “yes” for almost Delaunay triangulation. Statistically we found that if we select the destination nodes (heads) from the central parts of the graph, the possibility that the compass routing will guarantee the delivery of information to the head is much higher.

### 3.11. Long Edges in Delaunay Triangulation

Edges in the Delaunay triangulation which are longer than the optimal transmission range  $r$  of the nodes are called the long edges. Long edges in the Delaunay triangulation can occur in two places.

1. Edges that have at least one vertex on the convex hull or both the vertices are very near to the convex hull.
2. Near the big hole in the sensor network.

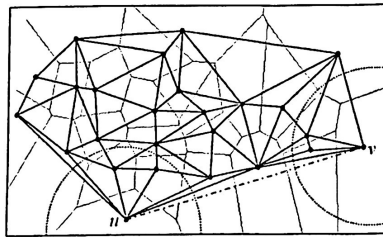


Figure 3. Long edge in Delaunay triangulation.

Figure 3 shows an example of long edges. We have to take care of the long edges near the boundary. It can be shown very easily that two nodes  $u$  and  $v$  in figure 3 can not decide whether the edge  $uv$  will be included in the triangulation or not. We can see that the nodes  $u$  and  $v$  are very far compared to their transmission range shown in dotted circle. Node  $u$  and  $v$  can only decide whether the edge  $uv$  will be included or not in the Delaunay triangulation if they have the information about all the nodes in the network. Delaunay triangulation is not possible to generate locally because of these long edges. Big holes can be handled by compass routing along with perimeter routing described in [16].

#### 4. Almost Delaunay Triangulation Routing Protocol

As Delaunay triangulation is not possible to generate locally. We introduce almost Delaunay Triangulation. Almost Delaunay triangulation is a subgraph of the Delaunay triangulation. We define the almost Delaunay triangulation as the Delaunay triangulation without the long edges. Note that, almost Delaunay triangulation is sometimes not even a triangulation like in Figure 4. Figure 4 shows the almost Delaunay triangulation of the triangulation in Figure 1. Long edges are displayed in thick dotted line and will not appear in the almost Delaunay triangulation. We can see that without the long edges the graph is not a triangulation. It can be shown that the nodes can locally decide which edges are in the almost Delaunay triangulation. This graph could be generated and used very efficiently to route the sensed information.

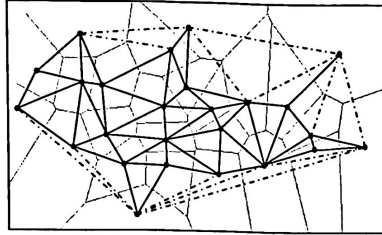


Figure 4. Almost Delaunay triangulation.

Routing in sensor network or ad-hoc network based on Delaunay triangulation is studied in [2][3][7][14][24][27]. It is well known that the Delaunay triangulation cannot be generated locally. In this paper, we propose an almost Delaunay triangulation routing protocol for wireless sensor networks. After deploying the sensor nodes in the target area, there is no established network among the sensor nodes. The sensors do not have any idea about the location of the other nodes and the architecture of the network. In the proposed approach, each sensor node will perform the following tasks after deployment.

1. Gather neighbor lists.

2. Locally generate the almost Delaunay triangulation.
3. Select head in each round and send the head's information to all the nodes.
4. Route the information using the almost Delaunay triangulation in a greedy approach.
5. Change the head after some predefined time interval.

At first all the nodes should gather information about its neighbors. After deploying the nodes, each node will send a *hello* message with some energy that will reach up to the distance  $C \times E_{\text{MSTMAX}}$ .  $E_{\text{MSTMAX}}$  is the longest edge in the minimum spanning tree of all the sensor nodes and  $C$  is some predefined constant. Here we assume that  $C \times E_{\text{MSTMAX}}$  is less than or equal to  $r$  (communication range of the nodes). Initially the nodes do not have any idea about the value of  $E_{\text{MSTMAX}}$ . The nodes could approximate the value of  $E_{\text{MSTMAX}}$  if they have prior knowledge about the target area  $A$  and the number of nodes  $N$ . When a node sends *hello* message, the neighbors within the distance  $C \times E_{\text{MSTMAX}}$  will receive the message. Then the neighbors will send an acknowledgement message along with its location information back to the sender. Figure 3 shows the neighborhood of node  $u$  and  $v$  with circle of radius  $C \times E_{\text{MSTMAX}}$ . In Figure 3 the value of  $C$  is 1. Thus each node will get the list of all its neighbors. After all the nodes have got their neighbor lists, we will get a connected graph.  $C \times E_{\text{MSTMAX}}$  will ensure that the graph will be connected if  $C \geq 1$ . Here each node in the graph only knows about its position and the position of its neighbors. Nodes do not have any other information about the remaining part of the huge network.

Reference [5] proposed a distributed algorithm to generate the Delaunay triangulation. In [5] all the nodes require to exchange a lot of information among the nodes which is not feasible for sensor networks. In the sensor network we cannot afford to exchange so much information, because it will be very much costly in terms of power consumption. So, we make some relaxation in the Delaunay triangulation. We propose an algorithm to generate the almost Delaunay triangulation by using the local information at each of the nodes. In this case we don't have to exchange any information among the nodes that are far away in the network. Note that almost Delaunay triangulation is not always a triangulation. After the almost Delaunay triangulation is generated all the nodes now know about its neighbors in the graph. For the remaining of the network we will use the neighbor information for routing. Now our task is to select a head for each of the rounds. In our approach we use probability to select the head. Our probability selection algorithm is much similar to that of LEACH [25] with some modifications. Our head selection is discussed later.

Once the almost Delaunay triangulation is generated, all the nodes will have to only remember the neighbors in the almost Delaunay triangulation. On an average each node has 6 neighbors [1]. So, each node has to store the information about 6 adjacent nodes. Now the sensing task will be divided into rounds. In each round, the head is selected and the head's location information is sent to all the nodes. Then each node will choose one neighbor in the closest direction to the head for routing. This routing is known as compass routing [3]. In compass routing only local information and the information about the destination is used to route the information. No other external information is required for routing.

#### 4.1. Algorithm

We can generate the almost Delaunay triangulation by the following steps.

**Step 1:**

(a) Send hello message to the distance  $C \times E_{\text{MSTMAX}}$  and wait for the reply. Here,  $C$  is some constant.  $C \geq 1$  will ensure that the graph will be connected. Approximation of  $E_{\text{MSTMAX}}$  is shown in section 8.

(b) Upon receiving hello message, acknowledge the sender along with the location information of the current node.

(c) Update the neighborhood list according to the acknowledgements of the hello messages.

**Step 2:**

For each node  $u$ , calculate the Voronoi polygon formed by the perpendicular bisectors of each line segment  $(u, v_i)$ , where  $v_i$  is the neighbor of  $u$ . To generate the almost Delaunay triangulation, we will connect  $u$  with the neighbor nodes that has common edge in the Voronoi polygon. The algorithm is very simple. After the execution of the algorithm each node will need to remember only the neighbors that have edge with it in the graph and forget the other neighbors.

#### 4.2. Head Selection

For each round we should select the head. For simplicity, we assume the whole network is a single cluster. Our head selection algorithm is much similar to LEACH [25]. Each node assigns itself some probability to be the head in the next round. This probability depends on the following criteria.

1. Remaining battery power.
2. Location within the network.

LEACH [25] only considers the remaining battery power to assign the probability. If a node has much remaining battery power then it is more likely to be the head. In our approach, if a sensor is near the convex hull or near a big hole then it will assign itself lower probability of becoming head. Nodes in the middle part of the target area will assign higher probability to select itself as head. Nodes could know whether they are near the boundary or not by observing the distribution of its neighbors. Here, we use a token based head selection algorithm. The token is generated by the base station, which is very far away from the target area. Base station does not have any problem of power and it knows about the target area and the distribution of nodes in the target area. At first the base station sends the token to any arbitrary node in the target area. The token contains a probability variable, which is the minimum required probability for being the head. If the receiving node has higher probability then the minimum requirement then it will select itself as the head and will inform that it is the head for the current round to the base station. Base station will then broadcast a message about the location of the head of the current round to the whole network. If its probability is less than the minimum requirement, then it will send the token to one of its neighbors. After each round, all nodes will update their probability accordingly, as their battery power will decrease after each round. At the end of a round the head will

reset the minimum requirement of becoming the head in the token and will release the token to one of its neighbors.

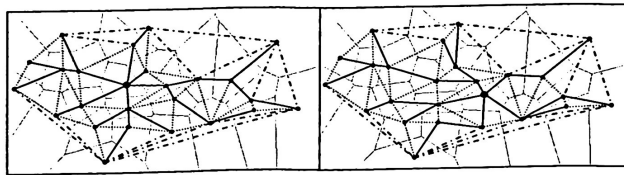


Figure 5. Routing tree in two different rounds.

Figure 5 shows the topology of the sensor network in two different rounds by using the compass routing in the same almost Delaunay triangulation. Head in each round is displayed in larger solid circle and the routing tree is displayed in solid lines.

#### 4.3. Initial Head Problem

In the above mentioned head selection approach, there is a problem. We have mentioned that at the very beginning (just after the almost Delaunay triangulation graph is generated) the base station will send the token to a sensor node randomly. However, the base station does not know the location of any of the nodes initially. For this reason, the base station will not be able to send the token to the initial head randomly. As the location information is used for addressing in sensor network, the base station needs to know the location of the initial head. To solve this problem, few (at least one) of the sensor nodes should send their location information to the base station. The nodes that will send their location information are the potential heads for the first round. The number of potential heads for the first round should be few. Because, sending the location information to the base station will need much energy. We have to select only few of the sensor nodes locally by the sensor nodes. Here we could use the degree information to select the initial potential heads.

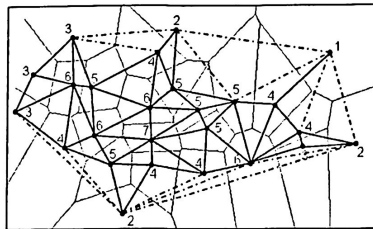
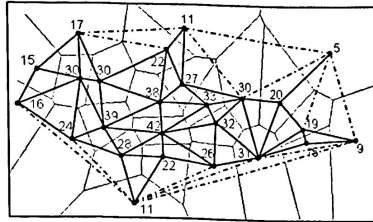


Figure 6. Degrees in almost Delaunay triangulation.

Figure 6 shows the degree of each of the nodes in the almost Delaunay triangulation. Some predefined integer *DEG* value will be used to select the initial potential heads. The nodes that have degree equal to or more than *DEG* will send their location information to the base station. However in this approach many of the nodes will be selected as we can see from figure 6 that many of the nodes has degree 5 or more. Instead of only using the degree of the current node we will use the degree of the current node and the degree of the neighbors of the current node. Figure 6 shows the degree of the current node plus the degree of its neighbors of the same graph.



**Figure 7. Degrees of current node and the neighbor nodes.**

Now we can use these values to select the initial heads. The value of *DEG* should be selected in such a way so that only a few but at least 1 initial potential head is selected.

## 5. Complexity

The algorithm to generate the almost Delaunay triangulation will run in  $O(n \log n)$  time. The complexity of the algorithm depends on step 2 to calculate the Voronoi polygon. The Voronoi polygon can be calculated in  $O(n \log n)$  time in the same way the 2D hidden surface removal algorithm works, simply by using divide and conquer method. As hidden surface removal algorithm is a well known algorithm, we will not cover the details about the complexity of generating the Voronoi polygon. We will just show that the generation of Voronoi polygon is the same problem as hidden surface removal problem. In 2D hidden surface removal problem, there are several lines and we have to find out the lines that are visible from a particular point. In Voronoi polygon generation, the current node knows the position of its neighbors. So, it will find the perpendicular bisector of each of the lines connecting the current node with its neighbors. Then the Voronoi polygon will consist of the perpendicular bisector lines that are visible from the current node. Hence, Voronoi polygon generation is the same as hidden surface removal problem.

## 6. Analysis

For any  $p_i$  and  $p_j$  in a finite set  $P = \{p_1, \dots, p_n\}$  of distinct points,

$$D(p_i, p_j) \leq c \times d(p_i, p_j) \dots \dots \dots (1)$$

holds [25], where  $d(p_i, p_j)$  is the Euclidean distance between  $p_i$  and  $p_j$ ,  $D(p_i, p_j)$  is the shortest path length in the Delaunay triangulation and

$$c = \frac{2\pi}{3 \cos \frac{\pi}{6}} \approx 2.42$$

This bound of  $c$  is not tight. In reality we will get a much smaller value of  $c$  in most cases. Equation (1) is also true for almost Delaunay triangulation but the value of  $c$  would be bigger than that of Delaunay triangulation. Statistically it is found that the value of  $c$  lies between 1.5 and 2 even for almost Delaunay triangulation in most of the cases.

In equation (1)  $D(p_i, p_j)$  indicates the minimum length path between  $p_i$  and  $p_j$ . However, compass routing does not always find the minimum length path. Theoretically it could be shown that the path found by compass routing is not even bounded by any constant  $c$  with respect to the Euclidean distance between  $p_i$  and  $p_j$  [2]. However, in practice the situations like this is very rare and this situation arises in highly degenerate case where the nodes are on the boundary of a single circle. We assume that the distributions of the nodes are uniform and random and thus we avoid the worst case. In our sensor network problem, Euclidean distance of the path found is not so important. We want to minimize the sum of the squared distance of each edge in the path. So, even though  $D(p_i, p_j)$  is almost 2 times longer than the Euclidean distance, the square distance is much better in the case of almost Delaunay triangulation. Delaunay triangulation ensured optimal squared distance as Delaunay triangulation maximizes the minimum angle in the triangles and so does almost Delaunay triangulation.

It may seem that randomly selecting some of the neighbor nodes will generate the same affect as the almost Delaunay triangulation. However, if we select the neighboring nodes randomly, the resulting graph will not be a planar graph; hence it will not support perimeter routing [16] in case of big hole described in section 3.

## 7. Experimental Results

In our algorithm, we are making a greedy decision at each node. This greedy approach does not ensure that we will finally reach to the head with the minimum cost path. So, in this algorithm the path found to the head will be higher than the  $D(p_i, H)$ , where  $H$  is the head. Table 1 shows the statistical result found in 500 routing trees in 10 different almost Delaunay triangulations. In our experiment we have used square target area  $A$  of size  $200 \times 200$  and number of sensor nodes  $N$  is 50. We have used uniform random distribution of the nodes in the target area  $A$ . The neighbors of the nodes are within the range  $C \times E_{MSTMAX}$ . Here we have used  $C=1.5$ . We did not approximate the value of  $E_{MSTMAX}$  here. We took the exact value of  $E_{MSTMAX}$ . After generating the almost Delaunay triangulation graph we have generated 50 rounds

(with all the 50 nodes as head) for each case. We have selected different heads in each round. The last column in Table 1 shows the number of cases where the compass routing was not successful, i.e. some of the nodes could not reach the head by using compass routing.

Table 1. Number of failures in almost Delaunay triangulation by using compass routing.

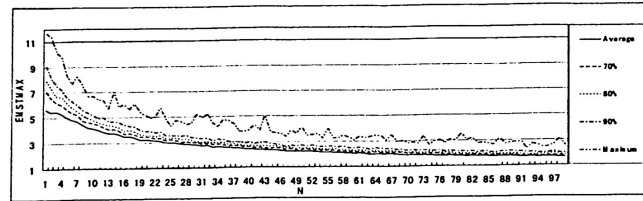
| #  | $E_{MSTMAX}$ | No. of edges in<br>Delaunay Triangu-<br>lation | No. of<br>long edges | Compass Routing<br>failed for # of heads |
|----|--------------|------------------------------------------------|----------------------|------------------------------------------|
| 1  | 96.02        | 137                                            | 7                    | 0                                        |
| 2  | 73.36        | 134                                            | 15                   | 6                                        |
| 3  | 79.06        | 132                                            | 7                    | 0                                        |
| 4  | 74.52        | 131                                            | 12                   | 1                                        |
| 5  | 86.09        | 136                                            | 11                   | 0                                        |
| 6  | 83.57        | 134                                            | 13                   | 0                                        |
| 7  | 98.81        | 134                                            | 5                    | 1                                        |
| 8  | 103.79       | 134                                            | 4                    | 0                                        |
| 9  | 75.74        | 134                                            | 12                   | 3                                        |
| 10 | 83.01        | 134                                            | 10                   | 0                                        |

In Table 1, we can see that compass routing is possible in most of the cases. Only for a very few times some of the nodes could not reach to the head by using compass routing in almost Delaunay triangulation. It is found from the experiment that these situations arise when we select the head near the convex hull or near the boundary of big hole. In our algorithm we assign small probability for the nodes near the convex hull or big holes to become the head and thus we avoid the situations like these.

## 8. Approximating $E_{MSTMAX}$

The nodes in the target area do not know about the value of the  $E_{MSTMAX}$  beforehand. We did some experiment to find the value of  $E_{MSTMAX}$  on random points. In our experiment we have used an area of  $10 \times 10$  unit and put  $N$  nodes randomly in it, for  $N=2,3,4,\dots,100$ . For each value of  $N$  we have generated 255 different random point sets. For each case we generated the MST and took the value of  $E_{MSTMAX}$ . The values of the  $E_{MSTMAX}$  are shown in figure 8. The value of  $E_{MSTMAX}$  varies largely for small  $N$ . But it is stable for bigger  $N$ . Figure 8 shows that the value of  $E_{MSTMAX}$  is stable for 90% of the cases for big  $N$ .

We could make these types of simulation in a computer to get the value of  $E_{MSTMAX}$  for a given area  $A$  and  $N$ . This preprocessed value of  $E_{MSTMAX}$  will be informed to all the nodes before deploying, which will be helpful to define the neighborhood.

Figure 8. Value of  $E_{MSTMAX}$ .

## 9. Error Handling

From table 1, we can see that sometimes the compass routing fails for almost Delaunay triangulation. This problem can be solved by perimeter routing alone with compass routing. Perimeter routing alone with compass routing was extensively studied in [16]. Whenever there is no neighbor closer to the destination (compass routing is not possible), then compass routing is switched to perimeter routing. Perimeter routing requires that the underlying graph is planar. Almost Delaunay triangulation is a planar graph and hence, it supports perimeter routing describe in [16]. During perimeter routing, if any intermediate node finds that it is possible to switch to compass routing, then it will switch from perimeter routing to compass routing. In this way we could solve the failure cases found in table 1.

Moreover, the value of  $E_{MSTMAX}$  could be arbitrary larger than our experimental value shown in section 8. Even though this situation is rare, we should handle this situation carefully. If  $E_{MSTMAX}$  is larger than the approximated value then the graph will be disconnected. However, we want the graph to be connected. After the graph is generated, some of the nodes will try to count the number of nodes in the network by using flooding of messages. Once the flooded message reaches some leaf node it will bounce back to the head through the same path. On its way back to the head, it will use a counter to count the number of nodes on the path. The head will sum up the counter of all the bounced messages. If the number of nodes is less than  $N$ , this means the graph is disconnected. So, all the nodes will increase their transmission range to discover some new neighbors in order to make the graph connected. This process will continue until the whole network becomes connected. This process is expensive, as we have to exchange a lot of messages among the nodes. Note that this situation is very much unlikely to happen in nature.

## 10. Conclusion and Future Work

We have used only one cluster. Further research could be continued to implement the same algorithm with multiple clusters. Multiple heads could possibly reduce the failure cases of compass routing in almost Delaunay triangulation and power consumption could be further reduced in this way.

## References

1. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, "Spatial Tessellations: Concepts and applications of Voronoi diagrams". Second Edition. Wiley Series in Probability and Statistics, 2000.
2. Chun-Hsien Wu, Kuo-Chuan Lee and Yeh-Ching Chung, *A Delaunay Triangulation Based Method for Wireless Sensor Network Deployment*, Proceedings of the 12th International Conference on Parallel and Distributed Systems - Volume 1, pp-253-260, 2006.
3. Evangelos Kranakis, Harvinder Singh, and Jorge Urrutia. "Compass routing on geometric networks". Proceedings of 11th Canadian Conference on Computational Geometry, 1997.
4. G.J. Pottie, "Wireless Sensor Networks", ITW 1998, Killamey, Ireland, p 139-140, 1998.
5. Hiroaki Kato, Takayuki Eguchi, Masaaki Ohnishi, Shinichi Ueshima, "Autonomous Generation of Spherical P2P Delaunay Network for Global Internet Applications". Proceedings of the fourth International Conference on Creating, Connecting and Collaborating through Computing (C5), pp-184-191, 2006.
6. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," IEEE Communications Magazine, vol. 40, no. 8, pp. 102-116, August 2002.
7. Jorge Urrutia, "Local solutions for global problems in wireless networks". Journal of Discrete Algorithms, 2007.
8. Karp, B. & Kung, H. T. "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks". MobiCom'00: Proceedings of the 6th annual international conference on mobile computing and networking. ACM press, 2000, pp 243-254.
9. Katayoun Sohrabi, Jay Gao, Vishal Ailawadhi, and Gregory J. Pottie, "Protocols for Self-Organization of a Wireless Sensor Network", Personal Communications, IEEE, Vol. 7, No. 5, pp. 16-27, 2000.
10. M. Keil and C. Gutwin, "The Delaunay triangulation closely approximates the complete Euclidean graph", Proceedings of the 1st Canadian Workshop on Algorithms and Data Structures, Ottawa, August 1989.
11. Madan, R. and Lall, S. 2006. "An energy-optimal algorithm for neighbor discovery in wireless sensor networks". Mob. Netw. Appl. 11, 3 (Jun. 2006), 317-326.
12. Martin Gruber and Günther R. Raidl, "Neighbourhood searches for the bounded diameter minimum spanning tree problem", 18th Mini Euro Conference on VNS, 2005.
13. Md. Bahlul Haider and Kokichi Sugihara, "Almost Delaunay Triangulation Routing in Wireless Sensor Networks", 10th International Conference on Computer and Information Technology, Dhaka, Bangladesh, 2007.
14. P. Bosc, P. Morin, I. Stojmenovic and J. Urrutia. "Routing with guaranteed delivery in ad hoc wireless networks". Wireless Networking, 7(6):609-616, 2001.
15. Prosenjit Bosc and Pat Morin. "Online routing in triangulations". SIAM Journal on Computing, 33(4):937-951, May 2004.
16. Qing Fang Jie Gao Guibas, L.J. "Locating and bypassing routing holes in sensor networks", INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies.
17. S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M.B. Srivastava, "Coverage Problems in Wireless Ad-Hoc Sensor Network," Proc. IEEE Infocom, 2001.
18. S. Slijepcevic, M. Potkonjak, "Power Efficient Organization of Wireless Sensor Networks", ICC 2001, Helsinki, Finland, June 2001.
19. Soro, S. and Heinzelman, W. B. "Prolonging the lifetime of wireless sensor networks via unequaling", Proceedings of the 19th IEEE international Parallel and Distributed Processing Symposium (Ipdps05) -

- Workshop 12 - Volume 13 (April 04 - 08, 2005). IPDPS. IEEE Computer Society, Washington, DC, 236-2.
19. Soro, S. and Heinzelman, W.B., "On the coverage problem in video-based wireless sensor networks", 2nd International Conference on Broadband Networks, Vol. 2, pp-932-939, 2005.
  20. Stephanie Lindsey and Cauligi S. Raghavendra, "PEGASIS: Power-Efficient Gathering in Sensor Information Systems", IEEE Aerospace Conference Proceedings, Vol. 3, pp-1125-1130, 2002.
  21. T. Rappaport, "Wireless Communications: Principles & Practices", Englewood Cliffs, NJ: Prentice-Hall, 1996.
  22. Tilak, S., Abu-Ghazaleh, N. B., and Heinzelman, W. "A taxonomy of wireless micro-sensor network models", Mobile Computing and Communications Review, 9 (2002), pp-28-36.
  23. Vangelos Kranakis, Harvinder Singh, and Jorge Urrutia. "Compass routing on geometric networks", In Proceedings of 11th Canadian Conference on Computational Geometry, 1999.
  24. Wendi B. Heinzelman, Anantha P. Chandrakasan, and Hari Balakrishnan, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks", IEEE Transactions on Wireless Communications, Vol. 1, No. 4, pp-660-670, 2002.
  25. Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. "Energy-efficient communication protocols for wireless microsensor networks". In Proceedings of the Hawaii International Conference on Systems Sciences, January 2000.
  26. Yu Wang and Xiang-Yang Li. "Efficient Delaunay-based Localized Routing for Ad Hoc Sensor Networks", Intl. Journal of Communication Systems, 2006.
  27. Zhong, L.C., Rabacy, J.M. and Wolisz, A., "Does proper coding make single hop wireless sensor networks reality: the power consumption perspective", IEEE Wireless Communications and Networking Conference, Vol. 2, pp-664-669, 2005.