A Clustering Based Fault Tolerant Network Full Coverage Method in WSN

Mohammad Mehrani, Ehsan Bidokh, Mohammad Hasannejad, Ali Mojoudi

Department of computer Engineering, Shoushtar Branch, Gotvand College, Islamic Azad University, Gotvand, Iran
m.mehrani@iau-shoushtar.ac.ir

Department of Computer Engineering, Andimeshk Branch, Islamic Azad University, Andimeshk, Iran
bidokh@iauandimeshk.ac.ir

Department of Computer Engineering, Andimeshk Branch, Islamic Azad University, Andimeshk, Iran
hassannejad@iauandimeshk.ac.ir

Department of Computer Engineering, Dezful Branch, Islamic Azad University, Dezful, Iran
mojoudi@iaud.ac.ir

Abstract

The management of Wireless Sensor Networks (WSN) depends on the algorithms used to group network nodes into clusters and once the clusters have been determined the protocols used within them to decide how the nodes provide coverage of their area. Effective and efficient algorithms to manage clustering and coverage need to address several very important factors that significantly effect performance, these factors include: network lifetime, scalability and load balancing between the nodes. Different algorithms are designed with different factors in mind. For example one algorithm may be designed to prolong the network lifetime and another for ease of scalability. Furthermore the implementation of a particular algorithm can also affect its performance. For example the choice of nodes to be the cluster heads may improve the effectiveness of a clustering algorithm. In this paper we use our Fault Tolerant, Energy Efficient, Distributed Clustering algorithm for wireless sensor network (FEED) which achieves at a significant functionality in clustering. Based on the FEED a coverage algorithm is used by the nodes to provide coverage of their area. The proposed FEED based coverage algorithm leads meaningful energy saving for network nodes while satisfies fault tolerance in the network .The estimations show a significant increase in fault tolerance. Receiving almost all the region information while considering energy, lifetime and fault tolerance problems existing in the most of wireless sensor networks is satisfied by FEED base coverage method.

Keywords: wireless sensor network, cluster head, coverage, fault tolerance, lifetime

I. Introduction

New developments in the electronic, communication, hardware and low-power design technologies have led to creating very useful networks called Wireless Sensor Networks (WSN). Sensor networks usually consist of a different number of sensor nodes that are dispersed in a region of the network. All network nodes have wireless communications and send information about their region to a base station directly or via other nodes.
The nodes have limitations such as their energy resources, processing power and communication capability that will limit the lifetime and capability of the network. It is important therefore that the mechanisms used to control the network are cognisant of these limitations. Energy efficiency for example is one of the most important factors in a WSN and control mechanisms that are energy efficient will improve the life-time of a network. But energy usage can also be impacted by other mechanism such as routing where the energy of a node is being reduced from routing requests. Therefore a routing protocol that reduces routing requests on a node will not only prolong its life-time but also the life-time of the network.

Along with each region in that a kind of wireless sensor network is implemented an enormous amount of data is available. Obviously transmitting all region’s data to the base station is not required such that this leads over energy consumption thus aggregating data in small groups and sending then to the base station can be assumed as a solution. (Mostafa, 2014, Kamali, 2014).

Clustering is an approach where all network nodes are organized into groups, called clusters, and communicate with a node that manages the cluster, called the cluster head. The cluster head is responsible for controlling the functionality and performance of its cluster. Part of its role is to collect the data sent by the nodes in its cluster and forward this to a base station. Usually there is one cluster head in each cluster but it is sometimes possible for clusters to have more than one cluster head in special applications. (Mehrani, 2011, Mojoodi, 2011) are our earlier works reviewed, mentioned and combined to achieve at the goals of this work. (Fu, 2011, Mostafa, 2014, Kamali, 2014, Asheer, 2014, Fei, 2011) are some other clustering and coverage methods for wireless sensor networks.

Some of the routing methods are power-aware meaning they consider the remaining energy during routing and other related activities as an important factor (Ettus, Lin, Meng, Singh, 1998, Shepard, 1996).

In some clustering algorithms (Heinzelman, 2000) cluster heads are selected randomly in every round, other algorithms (Younis, 2002, Gupta, 2005) are sensitive to factors such as energy and distance. Algorithm (Gupta, 2005) uses other techniques including fuzzy logic to form clusters. Selecting suitable cluster heads can reduce overall energy consumption and increase network lifetime. These are two important aspects for all WSN algorithms.

The authors of (Younis, 2002) divide clustering algorithms in three separate groups: some of them try to pay attention to network connectivity in parallel with controlling the transmission power level on all the network nodes aiming at increasing network capacity (Kawadia, 2003, Narayanaswamy, 2002). Power optimization is the main purpose in the second group clustering algorithms (Intanagonwiwat and Chang, 2000, Kulik and Heinzelman, 2002). The third group includes those of clustering methods who focus on the network topology controlling by denoting which sensors should be awake and which ones can sleep (Cerpa, 2002, Xu, 2001, Chen, 2002). Some of the clustering methods are also routing methods (Kawadia, 2003, Heinzelman, 2002, Lin, 1997, McDonald, 2001, Gerla, 2000) while some other are not (Banerjee, 2001, Basagni, 1999, Chatterjee, 2002, Kwon, 1999, Bandyopadhyay, 2003, Amis, 2000).

In this paper first we briefly reintroduce our earlier work as an energy-efficient clustering method (FEED) (Mehrani, 2011) that, when making clusters, considers energy, density, centrality and node proximity when choosing a suitable cluster head to manage the cluster. Then at the rest of the paper we review related works as material and methods. After that in section 3 we describe our coverage method based on the FEED platform. Section 4 reveals experimental
results of the coverage method. Finally we provide a summary of the contribution of the proposed approach.

II. Clustering Method

A. Brief Reintroducing Clustering Algorithm (FEED)

In FEED (Fault tolerant, Energy Efficient, Distributed Clustering) all network nodes are divided into clusters such that at the end of the algorithm there are some cluster heads (CH), some pivot cluster heads (PCH), and some supervisor nodes (SN). A CH node is a regular cluster head which is the head of its cluster. A PCH node is a pivot cluster head with additional capabilities beyond a CH node. All the PCH nodes together cover a big area of the entire network and are also the best nodes for acting as routers. It means that pivot cluster heads rather than their usual cluster head tasks can be used as the main routers for inter and intra cluster routing because they are accessible to all the network nodes directly or via ordinary cluster heads. Also, PCH nodes are the most powerful nodes with most remaining energy. A SN node is a supervisor node for its cluster head (CH or PCH) and will replace its CH or PCH when the CH or PCH fails. So, SN nodes are substitutes for their cluster heads and assist in the development of a fault tolerant clustered network.

There are four important factors in FEED that may have weights or use fuzzy logic to count their average contribution. These factors are:

A.1 Density
The density factor of a node specifies the number of nodes around it such that their distances are less than a threshold Dist1. It is very desirable to choose cluster heads from nodes that have high density factors.

A.2 Centrality
It is desirable to choose cluster heads from those nodes at the centre of their neighbours. Cluster heads shouldn’t be chosen from nodes at the border of the network because border cluster heads create problems for their members during communication. Sometimes a node has a high density factor meaning that there are lots of near nodes around it but they are all on one side of the node. In this case the centrality factor will reduce the likelihood it will be selected as a cluster head.

A.3 Energy
It is clear that cluster heads should be chosen from those nodes with enough remaining energy.

A.4 Proximity to other nodes
As mentioned, all nodes contribute to the choice of the cluster heads. A node that is going to be a cluster head is called a Volunteer. All nodes vote for Volunteers. A regular node should vote for its nearest Volunteer. Cluster heads are then selected from the Volunteer nodes that most nodes elect as their nearest one.

B. How FEED works

B. 1 Phase 1:
By the end of this phase each node will have its calculated density factor \((d_e)\). It should be noted that if all nodes are aware of their densities this phase can be omitted however after several rounds, some nodes will die and consequently node densities will change. Phase 1 executes just once in several iterations of algorithm execution. Each node is equipped with GPS and will be aware of its coordinates. Note that assuming GPS equipment for all the network nodes doesn’t lead much energy losing because it happens once in the network implementation. Also, this is usual for lots of other methods to assume GPS equipment for nodes.

For calculating densities, all nodes send their id and geographical coordinates to other nodes close to it by broadcasting locally. It is clear that local broadcast doesn’t need so much energy because in our work the broadcasting range is exactly calculated dynamically and won’t be so much, meaning that just the neighbour nodes in this case broadcast, locally. Nodes can broadcast their messages according to geographical coordinates in different time slices to avoid overhead problems. By knowing its own coordinates and that of the sender node, each node then computes its density factor \((d_e)\).

If the computed distance is less than \(D_{st1}\) the node increases its density field. At the end of this phase all nodes would know their densities.

**B. 2 Phase 2:**

In this phase if the density of a node is more than a threshold amount and also if it isn’t a border node, it computes its centrality factor \((c_e)\). Note that we do not let the border nodes to try to be cluster head. On the assumption that \(n\) is the number of network nodes and by using the following algorithm the centrality factor can be computed.

**B.2.1 Proposed algorithm for computing centrality (centrality algorithm)**

Consider figure 1 in which the red node at the centre of the figure is a regular node which has received local broadcast messages from the black nodes (other regular nodes). The distance between them is less than \(D_{st1}\). Also there would be a specific angle for each couple of red and black nodes that can be computed easily. Each black node also sends its id and coordinates in a local broadcast message. The angle between red and black nodes is computed from:

\[
\theta_{NodeID} = \tan^{-1}\left(\frac{(NodeID.Y-\text{me}.Y)}{(NodeID.X-\text{me}.X)}\right)
\]

Thus for each black node, the red node calculates its angle and increments the corresponding element in an array of 24 elements. According to figure 2, element 1 is for angles between 0° to 15°, element 2 is for angles between 15° to 30°, . . . and consequently element \(i\) is for angles between 15\((i-1)\)° to 15\(i\)°.

First all array elements are set to zero. The corresponding element is incremented when the angle is computed. The best case occurs when elements \(i\) and \(i+12\) are the same. It means the best case for centrality field is when the absolute value of difference between elements \(i\) and \(i+12\) equals zero. We save these absolute differences in array \(C\) as follows:

\[
C(i) = |B(i) - B(i + 12)|
\]

Note that the memory problem is almost solved in new sensor technologies. Also, few memory is needed which can be calculated. Then we sum all elements of array \(C\) and save it in variable \(c_e\) as the centrality factor. A smaller value of \(c_e\) creates a better situation for the centrality factor. So the best case for the centrality factor occurs when it equals zero. In this condition the node is
exactly at the centre of its neighbours. We give a negative sign to ce when we want to compute the average value of all factors.

![A regular node and its neighbours](image1)

Figure 1. A regular node and its neighbours

![Array B for saving angles](image2)

Figure 2. Array B for saving angles

All nodes are aware of their remaining energy (en). In continuation of this phase each node gives itself a score according to ce, de and en factors as follows:

\[
y = (a \times en) + (b \times de) - (c \times ce)
\]

(5)

a, b and c are coefficients for giving weights to energy, density and centrality factors, respectively. If the node is located on the border, it cannot be a volunteer. Taking this and all factors into consideration the node decides whether or not to be a volunteer. Then each volunteer sends a message consisting of y, co and id as follows:

\[
y \text{, local, broadcast, msg, \{myNodeID, co, y\}}
\]

(6)

**B. 3 Phase 3:**

All the nodes are aware of the messages from their neighbours mentioned by “(6)”. The messages received consist of y, co and id of the volunteers. Each node enters the distance factor to other factors for each volunteer and computes a second score for that volunteer. The second score is:

\[
z = (d \times y) - (e \times dis)
\]

(7)
d and e are coefficients for giving weights to y and distance, respectively. Each node gives a negative score to the distance factor to limit distant volunteers from being the cluster head for that node. For the rest of this phase, each node selects the neighbour volunteer with the best second score (z) as its deputy volunteer and reveals it to the other nodes around. The number of voters is very important for computing the final scores for volunteers. So, if a volunteer is selected as a deputy by several nodes, it will have a higher chance of being selected as a CH, PCH or SN. The way final scores are calculated is shown in table I [1].

TABLE I
GIVING FINAL SCORES TO VOLUNTEERS

<table>
<thead>
<tr>
<th>Volunteers' id and co</th>
<th>Given Scores by Other Nodes (given score • nodeid)</th>
<th>Num of Selectors (P)</th>
<th>Sum of Received Scores(S)</th>
<th>Final Score (F=P*S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x1,y1)1</td>
<td>(22,100)+(8,90)+(2,120)</td>
<td>3</td>
<td>310</td>
<td>930</td>
</tr>
<tr>
<td>(x3,y3)3</td>
<td>(17,98)+(4,100)</td>
<td>2</td>
<td>198</td>
<td>396</td>
</tr>
<tr>
<td>(x7,y)7</td>
<td>(43,143)+(33,112)+(44,123)</td>
<td>3</td>
<td>378</td>
<td>756</td>
</tr>
<tr>
<td>(x9,y9)9</td>
<td>(3,93)+(61,233)+(6,109)+(45,111)</td>
<td>4</td>
<td>546</td>
<td>2184</td>
</tr>
<tr>
<td>(x5,y5)5</td>
<td>(84,153)+(21,103)</td>
<td>2</td>
<td>256</td>
<td>512</td>
</tr>
<tr>
<td>(x4,y4)4</td>
<td>(12,196)</td>
<td>1</td>
<td>196</td>
<td>196</td>
</tr>
</tbody>
</table>

B. 4 Phase 4:
Now, each volunteer knows its final score and final scores of its neighbours as well so it can decide to be a CH, PCH, SN or even a regular node.

For making such a decision there are some rules for the location of CH, PCH and SN as shown in Figure 3. In Figure 3, we have: Dist1 <= Dist 2 <= 1.5* Dist 1

The most important rules in Figure 3 are as follows:
In every local area if the node has the highest final score it reveals itself as a PCH otherwise if the node doesn’t have the highest final score and also if its distance from nearest PCH is less than Dist1, it’ll check to know whether there isn’t any SN with a better final score for that PCH and by paying attention to this condition it introduces itself as a SN for that PCH.

But if its distance from nearest PCH is between Dist1 and Dist2 and if there isn’t any CH with a better final score for that area, it will try to exhibit itself as a CH. If there is any CH in this area but there isn’t any SN with a better final score for the nearest CH, the node introduces itself as a SN for that CH.

Another situation for the nodes is that its distance from nearest PCH is greater than Dist2, so if its distance from the nearest CH is also greater than Dist1, it’ll attempt to exhibit itself as a PCH, but if its distance from the nearest CH is less than Dist1 and in the situation that there isn’t any SN with a better final score for that CH, the node introduces itself as a SN for that CH.
In parallel with the above operations, regular nodes just listen to received messages from volunteers and distinguish nearest ones from others and finally join the nearest cluster head. Note that the lowest distance between every couple of PCH shouldn’t be less than Dist2, the lowest distance between every couple of CH shouldn’t be less than Dist1 and the lowest distance between any PCH and CH shouldn’t be less than Dist1.

When the remaining energy of a cluster head falls below the threshold amount, its SN would be replaced and that cluster can continue its operation until the end of that round. This feature of the FEED algorithm leads the region to be completely covered to the end of a certain round. Moreover, this feature leads to an increase in network lifetime and also leads to fault tolerance in clusters.

III. Result of the Clustering Method

To compare the LEACH, HEED and FEED algorithms, we simulated and executed them in the Matlab environment. We used three different network models: in the first model we dispersed 100 nodes randomly in a 100*100 region; in the second model we dispersed 200 nodes randomly in a 200*200 region; and in the third model we dispersed 300 nodes randomly in a 300*300 region.

All the initial conditions for the three algorithms mentioned were exactly the same. The radio model is shown in table 2. In the simulation we considered not only the Setup Phase but also the Steady State Phase in almost real conditions. In a Steady State Phase regular nodes receive (sense) data from the region and send them to their cluster heads. Each cluster head gathers and aggregates the received data and sends it to the Base Station. The Steady State Phase continues
until the remaining energy of at least a certain percentage of cluster heads falls less than the threshold. 
As mentioned before, when the remaining energy of a cluster head falls less than the threshold amount, its SN would be replaced and that cluster can continue its operation until the end of that round. This feature of the FEED algorithm leads the region to be completely covered to the end of a certain round. Moreover, this feature leads to an increase in network lifetime and also leads to fault tolerance in clusters.

Figures 4, 5, and 6 show the simulation results for the three models mentioned above in completely same conditions for LEACH, HEED and FEED algorithms. As shown, FEED algorithm increased network lifetime about six times (five hundred percent) more than the LEACH algorithm and about two times (one hundred percent) more than the HEED algorithm. Supervisor node replacement can be a reason for this enhancement.

Clustering algorithms should make clusters such that the number of single clusters (a cluster with only one member) is minimized. By applying the FEED algorithm we almost never see single clusters, so cluster distribution of this algorithm is suitable.

Figure 7 shows the percentage of total remaining energy of the network nodes after 1, 20 and 50 rounds. After one round, the HEED algorithm has the best outcome, but in later rounds the FEED algorithm has the best outcome. Note that after round 300 only FEED algorithm is still executing whereas the LEACH and HEED algorithms have terminated.

As a shortcoming of FEED, we can point to the necessary exchange of messages sent by nodes creating local overhead for the network. But fortunately this overhead is local and doesn’t cause huge problems.

**TABLE II**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Energy Dissipated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmitter Electronics</td>
<td>Eelec=50nJ/bit</td>
</tr>
<tr>
<td>Receiver Electronics</td>
<td>Eelec=50nJ/bit</td>
</tr>
<tr>
<td>Transmit Amplifier</td>
<td>100 pJ/bit/m2</td>
</tr>
</tbody>
</table>
Figure 4. Total number of nodes per rounds in the first model

Figure 5. Total number of nodes per rounds in the second model
IV. Coverage Method
We mentioned our earlier work (FEED) as the desired clustering algorithm for the platform of the coverage algorithm. In [29] we estimated a new redundancy based design for enhancing wireless
sensor networks routing. By mixing FEED and the work proposed in [29] we can achieve at fault tolerant network full coverage. This coverage idea works as following.

First the network will be clustered by FEED algorithm such that the clustered network would have some cluster heads: CH and PCH. As mentioned before, each cluster head has a SN which works as the substitute node and helps coverage as well. Generally the network would have several pivot nodes (CH and PCH) acting as the main nodes for coverage. Each regular node that senses data about it’s around region should send it to the base station. It performs this via pivot nodes. To achieve at full coverage all the information belonging to the all the network nodes should be received by base station. So for making sure about receiving information by base station correctly, the packets can be sent through more than one path. Thus the messages are sent through redundant paths such that both the fault tolerance and full coverage would be satisfied.

Like our earlier work we estimate different scenarios and calculate the results.

Clearly the network may have some faults during data transmission which varies from 0.001 and 0.1, and a kind of fault belonging to the sensor nodes which varies from 0.0001 and 0.001. In this scenario the number of correct replies of the system would be computed. There are 1 to 50 clusters needed in the network that affects the functionality of the algorithm. Also there are one, two, three or four redundant paths shown by m. As mentioned in [29] the reliability of a query that requires k clusters to respond is given by:

$$R_{qm}(k) = 1 - P_{km}$$

And the needed energy for sensing and transmitting nb bits is given by:

$$E_{packet} = 2n_bE_{elec}$$

The overall energy consumption existing in k clusters equals $$E_{packet} * N_h * k * m$$. Thus, the spent energy by the system, Eq(k), to answer a query that needs k clusters is given by: [29]

$$Eq(k) = E_{packet} * N_h * k * m$$

The average number of queries that the system is able to sustain before running out its energy is given by:[29]

$$N_q = \frac{E_{initial} - E_{threshold}}{Eq}$$

If the system is able to reply Nq requests with reliability factor Rq then the number of the requests that can be replied correctly before system loses its energy can be used as a scale to measure the network life time [29]. The results are shown in figure 8 to 12.
Fig 8. The Number of Correct Answers having Transmission Fault equals 0.001 & The Sensor Fault equals 0.0001
Fig 9. The Number of Correct Answers having Transmission Fault equals 0.005 & The Sensor Fault equals 0.0005

Fig 10. The Number of Correct Answers having Transmission Fault equals 0.01 & The Sensor Fault equals 0.001
Fig 11. The Number of Correct Answers having Transmission Fault equals 0.05 & The Sensor Fault equals 0.005

Fig 12. The Number of Correct Answers having Transmission Fault equals 0.1 & The Sensor Fault equals 0.01

V. Conclusions

In this paper we used FEED, a fault tolerant, energy efficient, distributed clustering algorithm and based on FEED proposed a coverage method that uses redundant paths for fault tolerant full coverage.
The Number of Correct Answer to Queries versus the Number of Required Clusters for Each Query has been considered as different scenario and shown in different pictures.

Acknowledgements

This paper presents the result of our research program belonging to the Department of computer Engineering, Shoushtar Branch, Gotvand College, Islamic Azad University, Gotvand, Iran.

References

58


16. O. Younis and S. Fahmy, “HEED: A Hybrid, Energy-Efficient, DistributedClustering Approach for Ad-hoc Sensor Networks”, sponsored in part by NSF grant ANI-0238294 (CAREER) and the Schlumberger Foundation


