Research Article

An Efficient Nodes Failure Recovery Management Algorithm for Mobile Sensor Networks

Rab Nawaz Jadoon,1,2 Adnan Anwar Awan,3 Muhammad Amir Khan,3 WuYang Zhou,1 and Aamir Shahzad3

1Key Laboratory of Wireless-Optical Communication, University of Science and Technology China, Hefei 230027, China
2Department of Computer Science, COMSATS University, Islamabad, Abbottabad Campus 22060, Pakistan
3Department of Electrical and Computer Engineering, COMSATS University, Islamabad, Abbottabad Campus, Pakistan

Correspondence should be addressed to Rab Nawaz Jadoon; rbnawaz@mail.ustc.edu.cn and WuYang Zhou; wyzhou@ustc.edu.cn

Received 17 March 2020; Revised 25 June 2020; Accepted 3 August 2020; Published 19 August 2020

Guest Editor: Hou-Sheng Su

Copyright © 2020 Rab Nawaz Jadoon et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Wireless sensor networks are not prone to harsh environments and may fail due to various reasons. Failure of sensor nodes causes partitioning of network into various small segments and restricts the communication of nodes. Due to the significant importance of restoration mechanisms, many approaches have been proposed in the literature so far. However, these approaches do not focus on uniform distribution of sensor nodes before the occurrence of failure. This paper fulfills the shortcoming in the literature by proposing a Uniform Distribution and Recovery Algorithm (UDRA) in two parts. The first part (prefailure algorithm) focuses on preparing the mobile sensor nodes to be ready for the failure beforehand by maintaining half of their communication distance between them. Also, it uses a novel method of directional matrix based on one-hop information. By using this method, each mobile node declares itself as cut-vertex (CV), intermediate node, or leaf node. The second part of the algorithm (postfailure algorithm) gives complete recovery procedure in the network by its recovery nodes. The extensive simulations prove that the proposed algorithm supersedes the existing approaches.

1. Introduction

Wireless sensor networks (WSNs) are an active area of interest for the researchers and engineering communities. A WSN is typically a network of distributed sensor nodes collecting data from the environment to monitor the physical conditions of that environment. The distributed nodes are of two types, namely, stationary nodes and mobile nodes. Stationary nodes remain stationary and perform sensing activities, routing of data, or act as sink nodes, whereas mobile nodes move around the network to perform sensing and different tasks like replacing failed nodes and may be equipped with different equipment such as fire extinguisher and guns to shoot enemies. These mobile nodes are known as actor nodes and such type of network is called Wireless Sensor and Actor Networks (WSANs). Actors have many capabilities, such as processing of sensed data, decision making, and performing suitable actions as mentioned above. Robotic Mule [1] is an example of such an actor designed for military use for detection of mines in a battlefield. These distributed nodes are connected to a sink node, which is further connected to the wired world for the usage of the data.

The data provided by the sensor nodes may be used for monitoring the area to make suitable decisions and strategies. For example, in agriculture, this data may be used to make strategies for controlling temperature for different crops or in military decisions this data may be used to make movements of troops and other military equipment according to the enemy strength along the border. These kinds of networks are most suitable for harsh environments in which awareness of the surroundings is critical without compromising human lives. WSNs are very effective for many applications in important domains such as security
systems, surveillance, battlefield reconnaissance, space exploration, search and rescue, and deep sea exploration [2]. The performance of WSNs totally depends upon the fact that the sensor nodes remain connected to the sink node for sending data, as there may be some nodes which are positioned far away from the sink node and need many hops through their neighboring nodes to transmit their data [3, 4] to sink node. Another important fact is that the sensor nodes must cover maximum part of the monitored area for the maximum awareness of the environment.

Achieving connectivity and coverage in an integrated manner is practically a challenging task. One of the challenges is the design of sensor nodes as sensor nodes are small, having limited resources for processing and storage. In order to maintain WSN, mainly batteries are used as the central source of power for these sensor nodes [5], as the batteries used have limited power and get drained with the passage of time resulting in failure of nodes. Another challenging issue is that, in harsh environment such as a battlefield, the sensor nodes may be destroyed by the explosives. Failure of a node or multiple nodes may affect the connectivity and may cause partitioning of the network into multiple disjoint segments, which restricts many nodes from transmitting their data to the sink node. This may also cause connectivity and coverage holes in the area, which compromises monitoring process of the environment. Ineffective monitoring of the environment may cause harm to the entire network or fail to achieve objectives of monitoring because an unattended danger might be present in the coverage hole. Hence, restoration of connectivity and coverage for the network is of utmost importance.

A solution to this problem is to deploy fresh nodes replacing the dead ones which is a costly and sometimes infeasible process. The restoration process must be self-healing, self-organized, and fault tolerant [6] which makes sensor nodes useful in scenarios with minimum or no human interaction. Generally, sensor nodes are deployed in a random manner which creates a highly connected network in some areas and leaves the other areas partially connected. As coverage area may be compromised in such situations, sensor nodes must be capable of adjusting themselves in such a way that their distribution in the area becomes uniform and maximum coverage is attained. Usually, the distribution is supervised by the sink node which can process data and make decisions, but this method increases the overall communication overhead as the decisions must pass the entire network in form of messages and incur delay. Therefore, it is highly recommended that nodes must arrange themselves to decrease the overall messages in the network.

Uniform distribution of actor nodes has already been discussed in [7], but this approach does not give the exact distance by which each node must separate itself from the neighboring nodes, which results in two possible scenarios. First is the isolation of a single node or multiple actor nodes from the entire network. The second situation is the isolation of a network or multiple networks from the rest of network. These two scenarios are not suitable for mission critical networks in a harsh environment like a battlefield and can cause disconnection of many nodes. To avoid these two scenarios, nodes must be cognizant of the exact distance by which they have to be separated from their neighbor nodes to achieve uniform distribution of nodes in the entire network. Based on our experimentation, we suggest that the distance should be half of the transmission range and nodes must be aware of this fact that they should maintain half of the distance of their communication range. For experimentation, we assume that all nodes have the same sensing and communication range. It is also assumed that transmission range and sensing ranges are also equal. All nodes are deployed in a random manner. Physical constraints are also taken into account in this paper, which are often not considered in many techniques in the literature so far. This paper proposes a new mechanism for the nodes to declare themselves as cut-vertex, intermediate, or leaf node by calculating point-to-point coverage area in all directions, i.e., east, west, north, south, north-east, north-west, south-east, and south-west; details are given later in this paper. After declaration, neighboring nodes are declared as recovery nodes, which are responsible for connectivity in case of failure. The main contributions of this paper are as follows:

1. UDRA focuses on proactive approach and stresses more on the prefailure operations rather than postfailure operations and is most suitable for mission critical applications which need very less time for connectivity and coverage restoration.
2. UDRA also considers the physical constraints, which are often not taken into account, hence giving a practical solution.
3. UDRA gives the exact distance between two nodes for distribution in area and all nodes try to maintain this distance. This gives advantage that these nodes stay connected in case of failure of any node without movement of such nodes as described earlier.
4. UDRA also gives the complete recovery plan, in case of a failure of single node, if some nodes or multiple nodes cannot maintain a distance Rc/2 between their neighbors due to physical constraints.
5. UDRA also gives a very simple solution to find cut-vertexes, intermediate, and leaf node by one hop information by node itself.

The rest of the paper is organized as follows. Section 2 gives the comparison of the proposed algorithm with the related work done in the past. Section 3 has the details of the proposed algorithm along with the pseudocode. Validation of results is reported in Section 4, and Section 5 concludes the paper.

2. Related Work

Preserving connectivity while maintaining overall coverage intact by using limited number of mobile nodes is a difficult task, especially when a single or multiple nodes fail. It is the problem of utmost importance which has been studied in the literature in recent years. Various solutions have been proposed regarding this issue. There are two kinds of
solutions. The first kind of solutions depends upon relocation on demand, whereas the second kind of solutions is based on postdeployment relocation. In general, sensor nodes are deployed randomly by aerial deployment. This kind of deployment may cause nonuniform distribution of mobile nodes in the AOI; i.e., some areas may have higher density and some areas may have lower density of mobile nodes. Therefore, relocation is desired in this situation so that the connectivity is assured between mobile nodes and the end user to maximize the coverage area. Connectivity issue has been targeted in the literature and many solutions have been provided in this regard [7–13]. To solve connectivity issue, coverage holes must be avoided in some areas. Therefore, a balance between connectivity and coverage restoration is important to keep both intact. Several approaches have been used in recent years.

In [8], the authors proposed a distributed algorithm for the network connectivity restoration, when a node is failed. For the restoration of network connectivity and improvement of coverage, the technique uses mobile nodes at the time of failure. A concept of 2-connected network was introduced in [13]. This means that among each pair of nodes there must exist at least two connected pathways in the network. This approach has 2-degree connectivity and requires 2-hop information for each node. In [7], coverage-aware and connectivity constrained actor positioning (C2AP) algorithm is proposed. In this approach, post-deployment of nodes is discussed. In [14], a cascade movement of nodes was given as a solution when some nodes fail. According to this technique, the neighboring node of the failed node replaces it. Then, such node is replaced by another node and replacements continue until being replaced by redundant node.

In [9], there is an algorithm known as Recovery by Invert Motion (RIM). In RIM, the recovery of the failed node was done by its neighboring nodes by invert movement towards the failed node. The main idea is that, through missing of HEARTBEAT message, neighbors will know about the node failure. Therefore, neighbors make inward movements to the failed node for restoration of the network connectivity.

Prefailure cut-vertex nodes determination is done by Partition Detection and Recovery Algorithm (PADRA), which is presented in [10]. Failure of cut-vertex nodes causes disjoints in the network. The suitable neighbor is selected for each cut-vertex node to handle its failure. Hence, in this way, prefailure connectivity and coverage are restored. The connectivity restoration process involves relocations of all the involved neighboring nodes, which causes cascaded movements of such neighbors.

In [11], coverage-aware connectivity restoration (C3R) in mobile sensor networks technique gives the solution by replacing the failed nodes by their neighboring nodes. The solution suggests the to and fro movement of the neighboring nodes between the position of failed nodes and their original position. The schedule of changing turns of these nodes was made by the node which reaches the failed node first.

To restore the connectivity of the network, an algorithm known as Volunteer-instigated Connectivity Restoration (VCR) is proposed in [12]. In this algorithm only, the immediate neighbors of the failed actor node take part. The selection criterion of the neighboring nodes is based on nearness. The cascaded movement is limited in this algorithm.

A provision of connected path planned by controlling a robot is described in [13]. Minimizing distance with minimal number of hops counts for mobile robots in such a way that connectivity remains intact which is the supreme goal of this algorithm. The proposed approach contains two algorithms. For allocation of a robot to an event location, the nearest robot to the event is identified by the first algorithm. Then, the nearest nonconnected robot is searched by this algorithm. The nearest robot is then directed to bring itself forward to connect to any of the joined section with allocated robot. This procedure is performed until the network becomes completely connected. For minimization of the hop count between the base station and the event area, the second algorithm is used. It optimally provides such locations where there are minimum numbers of hops.

However, a Backbone Polygon (BP) is built around the center of the disjoint network area in Connectivity Restoration with Assured Fault Tolerance (CRAFT) algorithm presented in [14]. To connect each outer partition to the BP, by two nonoverlapping paths, Relay Nodes (RNs) are positioned in the targeted area having low cost, resulting in the restoration of the connectivity of the network.

In [15], the authors give an exceptional connectivity restoration technique called Geometric Skeleton based Reconnection (GSR). GSR uses a geometrical skeleton based approach to partition network logically into different segments. A geometrical skeleton backbone consists of a group of nodes that have the maximum connectivity with other nodes. Each segment has record of all the skeletal backbone nodes, so, in case of network partitioning, each segment attempts to join the geometrical skeleton backbone. In this way, connectivity can be restored.

To observe the influence of realistic terrain, hybrid recovery strategy based on random terrain (HRSRT) is proposed in [16]. The Area of Interest (AOI) is divided into a grid of cells. All cells have equal size. Mapping of these cells is used to plan the terrain. The weight $\omega(c)$ represents each cell $c$. The weight of each cell $\omega(c)$ can be calculated by the summation of weight of each path $\omega(P)$. A complete graph $K_n$ is created by considering the minimum weights of paths between segments. A tour $T$ for MDCs (mobile data collectors) is done by a random terrain-based path planning algorithm, which is established on $K_n$ for initialization of connectivity restoration. The weight of tour $\omega(T)$ is directly proportional to the cost of data collection and aggregation. There may be one or more than one MDCs. This results in different relay nodes deployment approaches. So, for connectivity restoration, an "optimized relay node deployment" and "relay nodes deployment" are responsible. Minimal energy and cost for data aggregation and collection are the goals of these algorithms.

In [17], Survivability-Aware Connectivity Restoration (SACR) is given. Data load levels of disjoint parts are considered for connection of the partitioned segments.
between different disconnected segments. A set of moveable nodes are located in the proper location of these inaccessible segments. A relay partition is generated for every isolated segment, immediately after isolated segments are traced. This causes restoration of connectivity from the AOI to the base station.

Another technique for calculating cut-vertex is presented in [18]. For this purpose, this technique uses two localized and distributed algorithms. For the detection of cut-vertex and noncut-vertex nodes, a local subgraph having two hops with connected dominating set (CDS) knowledge is used in the first algorithm. The second proposed algorithm depends entirely on the first one. For unrecognized parts of the network, a limited distributed depth-first search algorithm is used without searching the whole network. This algorithm discovers the states of all nodes by comprehensive test bed experiments with the presence of a CDS. Cut-vertexes nodes are detected with lower energy consumption. Simulations results are the witness of this fact.

A method for maintaining confidentiality of location of roaming position-based services (PBSs) users using machine learning techniques is suggested in [19]. For roaming PBS users, the authors suggest a triphase process. It identifies user position by merging decision trees and \( k \)-nearest neighbor and estimates user destination using hidden Markov models along with position track sequence. In addition, the proposed model follows a mobile edge computing service strategy which will ensure the timely delivery of PBSs. The advantages of a mobile edge service policy offer confidentiality of position and low latency through networking and computing services in the vicinity of roaming users.

In [20], an energy-conscious green adversary paradigm for its use in intelligent industrial environments by maintaining confidentiality is proposed. Although the hardware and software parts of cyber physical systems can be mutually improved to reduce their energy consumption, this paper focuses on aspects of position conservation and confidentiality of the information. Based on the literature findings (assumptions, adversary targets, and capabilities), the authors include some testimonials to support cyber physical security practitioners and researchers. The proposed model is running on real-time, anticipatory position-based query scheduling to reduce the cost of communication and computation for each query, thereby encouraging minimization of energy consumption. In addition, to avoid degrading slots, we measure the transfer/acceptance slots that are needed for each question. The experimental findings indicate that, in comparison with current methods, the proposed solution will reduce energy consumption by up to five times.

The key objective explained in [21] is to extend the lifespan of the network by extending the lifespan of the working sensors as well as moving the collected data from the super node into the sink. Bat Algorithm (BA) is used in this article to pick the optimal monitoring sensor node and corresponding direction to minimize energy consumption. Results of simulation and comparison with other algorithms suggest the superiority of the algorithm proposed. The simulation results of the proposed algorithm show that the proposed algorithm was able to reduce the network’s power consumption and increase the network’s service life. The proposed algorithm may also outperform the comparable algorithms by 27 per cent on average.

In [22], the whale optimization algorithm (WOA) is used in this paper to solve the resource allocation (RA) problem in Internet of things (IoT) with the goal of optimum RA and to reduce the overall cost of communication between resources and gateways. There is a comparison of the proposed algorithm with other current algorithms. Results suggest that the proposed algorithm works properly. The proposed process, based on different metrics, is better than others in terms of “absolute cost of communication.”

The solution to this problem has three optimization objectives. They are as follows:

(a) Coverage restoration
(b) Connectivity restoration
(c) Both coverage and connectivity in an integrated manner

In order to achieve the above optimization objectives, three approaches are used.

(a) Reactive approach: in this approach, a recovery mechanism is developed after the occurrence of failure. References [9–11] are the examples of such approach.
(b) Proactive approach: in this approach, recovery mechanism is established before the occurrence of failure. It means that sensor nodes are prepared well before the failure. References [7, 18, 23] are the examples of such approach.
(c) Hybrid approach: in this approach, both reactive and proactive approaches are used. This approach is used according to the requirement. Yihani, Shivendra, 2006 [10, 24], etc. are the examples of such approach.

Table 1 shows optimization objective and nodes’ mobility and either algorithm is reactive, proactive, or hybrid and shortcomings of the techniques presented in the literature since 2007.

The proposed (UDRA) gives the solution to achieve connectivity restoration objective by using hybrid approach. The extensive simulations show that the proposed algorithm outperforms its baseline algorithms. Table 2 gives the details of optimization objectives, approach, and strengths of the proposed algorithm.

3. Details of the Proposed Algorithm

3.1. The Scenario and Assumptions of UDRA. The proposed algorithm considers all sensor nodes as mobile sensor nodes, deployed randomly in an Area of Interest (AOI). It is assumed that all nodes have capability of sensing the environment to find the physical constraints. This gives them the knowledge of their capability for the decision of maintaining \( R_s/2 \) distance with the neighbors. This can be useful for their availability to become recovery node for failure later. It is also assumed that, at the time of startup, all nodes discover
each other to form a network [8]. The proposed algorithm assumes that mobile nodes can determine their location relevant to their neighbors [9]. Each mobile node makes a list of its first hop neighbors, which reduces the messaging overhead in the network as compared to having two-hop neighbor information. This paper mainly focuses on proactive approach for the mobile nodes in such a way that it maintains connectivity as well as coverage. If the mobile nodes are not prepared before the failure, it will take some time to react, hence sacrificing the connectivity and coverage for a certain amount of time. URDA gives permanent relocation of mobile nodes in case of failure which assures both connectivity and coverage without wasting a lot of time.

3.2. Maintaining $R_c/2$ Distance Distribution. As mentioned in Section 1, the mobile nodes are deployed in random manner in some AOI. This may lead to a situation when there might be an abundance of mobile nodes in one area and shortage of mobile nodes in some other areas. Therefore, all mobile nodes must be uniformly distributed throughout AOI. To maintain uniform distribution, this paper suggests

<table>
<thead>
<tr>
<th>Table 1: Overview of related work algorithms.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>C2AP [7]</td>
</tr>
<tr>
<td>DCRA [8]</td>
</tr>
<tr>
<td>RIM [9]</td>
</tr>
<tr>
<td>PADRA [10]</td>
</tr>
<tr>
<td>C3R [11]</td>
</tr>
<tr>
<td>VCR [12]</td>
</tr>
<tr>
<td>DEEHC [13]</td>
</tr>
<tr>
<td>CRAFT [14]</td>
</tr>
<tr>
<td>GSR [15]</td>
</tr>
<tr>
<td>HSRT [16]</td>
</tr>
<tr>
<td>SACR [17]</td>
</tr>
<tr>
<td>CDS [25]</td>
</tr>
<tr>
<td>DCHS-WS [26]</td>
</tr>
<tr>
<td>MMLPP [27]</td>
</tr>
<tr>
<td>AUR [24]</td>
</tr>
<tr>
<td>TEEN [23]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2: Overview of the proposed algorithm.</th>
</tr>
</thead>
<tbody>
<tr>
<td>The proposed technique</td>
</tr>
<tr>
<td>--------------------------</td>
</tr>
<tr>
<td>UDRA</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
that these mobile nodes should try to maintain $R_c/2$ distance between them. If all the neighbors of a mobile node succeed in maintaining $R_c/2$ distance, then there are two advantages of this distance. The first is that all the mobile nodes will be uniformly distributed in AoI, as mentioned earlier. The second advantage is that, in case of some node failure, there is no need for the neighbors of the failure node to move towards that failure node and restore coverage and connectivity. The reason is that neighbors will stay connected to the immediate neighboring nodes of the failed node. The first advantage is obvious, as all nodes have the same distance between them, so the mobile nodes will be distributed in uniform manner. However, the second advantage can be explained by the example depicted in Figure 1. In the example shown in Figure 1, mobile nodes $A, B, C, D, E, F,$ and $G$ are successful to maintain $R_c/2$ distance between them. Half of the distance is taken by bisecting the distance between them by using dashed line segments just in case of failure of nodes $D$ or $F$. Their neighboring nodes $C$ and $A$ will be stayed connected with nodes $E$ and $G$, respectively. Their overlap coverage area, shown by shaded areas, remains intact. So, there is no need of movement for mobile nodes $C$ and $A$ for the restoration of connectivity and coverage. This saves extra movement in case of failure, hence saving extra energy spending on such movements, as the distance between the mobile nodes is well known by the localization methods as given in [9, 10]. The relationship between coverage area and distance is inversely proportional; i.e., the coverage area will be more when the distance between the nodes is less. The coverage area will increase with the decrease in distance. In Figure 2, it is depicted that there is the overlap area between nodes $N_1$ and $N_2$. $R_c$ is their communication range, whereas $\alpha$ and $\beta$ are the angles, which helps in solving the area under arc “$pq$” with centers $N_1$ and $N_2$. So, the area of the arc “$pq$” when center is $N_1$ can be calculated by the equations given in [15]

$$\text{area} = \left( \frac{\beta R_c^2}{\pi} \right) - \left( \frac{d R_c}{2} \right) \times \left( \sin \left( \frac{\beta}{2} \right) \right).$$ (1)

Equation (1) can also be used to find out the area under the same arc “$pq$” with the center $N_2$. Hence, the overall coverage area can be found by the equation [15]

$$\text{overlap} = 2 \times \frac{\text{area}}{\pi R_c^2}.\quad (2)$$

3.3. Determining Cut-Vertex, Intermediate, and Leaf Nodes.

For determining cut-vertex (CV), intermediate node (IN), and leaf node (LN), each mobile node is in a network. A novel method of directional matrix is proposed. All of the mobile nodes try to maintain $R_c/2$ distance between them. If all nodes maintain such a distance, then there will be no cut-vertexes. But still there is a chance that there might be some nodes which cannot maintain $R_c/2$ distance due to some physical constraints. In this case, the concerned node maintains a matrix showing the directions of its neighbors by their relative positions and the compass mounted on each node. By this matrix, a node declares itself as cut-vertex, immediate, or leaf node. This matrix is based on consisting of three rows and three columns as shown in

$$A = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}.\quad (3)$$

Here, in this matrix, the position of the concerned node is in the middle of the matrix and is given by element $a_{22}$, whereas the element $a_{23}$ shows number of one-hop nodes in east direction. The element $a_{21}$ shows number of one-hop nodes in west direction. The element $a_{21}$ shows number of one-hop nodes in north direction. The element $a_{32}$ shows number of one-hop nodes in south direction. The element $a_{11}$ shows number of one-hop nodes in north-west direction. The element $a_{31}$ shows number of one-hop nodes in south-west direction. The element $a_{31}$ shows number of one-hop nodes in north-east direction. The element $a_{33}$ shows number of one-hop nodes in south-east direction. Directions are mapped on matrix according to directions shown in Figure 3.

$$A = \begin{bmatrix} \text{nw} & \text{N} & \text{ne} \\ \text{sw} & \text{W} & \text{E} \\ \text{sw} & \text{S} & \text{se} \end{bmatrix}.\quad (4)$$

Initially, matrix $A$ has the following values:
i.e., 1 for the position of the concerned node and 0 for all other directions. As far as the concerned node finds its neighbors by overlap area, it finds their direction by its navigational compass and by their relative positions. The concerned node increments the number of first hop nodes in their respective directions. The cases of determining cut-vertex, intermediate, and leaf node can be best explained with the help of Figure 4. In Figure 4, consider node A; it has a node let’s say K, which cannot maintain the distance of $R_c/2$ with A. Node A has B in east, K in north, E at south, F at south east, and C at south west, so the directional matrix for A will be

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$  

(5)

Node A will broadcast the message with the IDs of its neighbors that it has B, C, E, F, and K with it at its first hop. Do they have any of these connected with them? So, all of the neighbors receive this message, and answer back to A with “NO CONNECTION” message, if none of them are connected to each other. So, A makes its value 0 in the matrix and checks for connections of its neighbors:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \quad (6)$$

As K has two nodes in north-east, i.e., N and L and two nodes in north-west, O and M. So, node K checks its neighbor connections by setting its value to zero:

$$K = \begin{bmatrix} 2 & 0 & 2 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \quad (7)$$

Similarly, all the nodes in the networks declare themselves as CV, IN, or LF. There might be a chance that there will be an IN and it declares itself as CV. After declaration,
3.4. UDR Algorithm

3.4.1. Prefailure UDR Algorithm. As mentioned in Section 1, nodes are deployed in random manner. So, there is a chance of profusion of nodes in one area and lack of nodes in the other. This rises to a coverage problem as all the area is not attended in uniform manner. Uniform distribution of nodes can be done by the sink node, taking instructions from the end user. But this type of strategy is not suitable as overall messaging overhead of the network increases, because sink node must transmit its messages to all the network nodes and must wait for their acknowledgment. So, nodes must rearrange themselves, without adding extra messaging overhead to the network. So, first, a node after deployment tries to find its neighbors by coverage overlap of their $R_c$. Then, each node looks around its environment to find out any physical constraints. If it finds any unavoidable obstacle, it just sets CM (cannot move) flag, by sending message to its neighbors along with its ID. Then, that node calculates directional matrix to find out either it is cut-vertex (CV), intermediate node (IN), or leaf node (LF) which is already discussed in detail in previous section. The neighbors update their neighboring list by putting CM and CV or IN or LF with that node’s ID. If there are no unavoidable obstacles, then a node compares its ID with neighboring node IDs. If its ID is greater than all other IDs, it sends message “distribution admin” (DA) along with its ID. After this, DA calculates each neighboring node distance, which may be less than, equal to, or more than $R_c/2$. If all neighboring distances are equal to the required distance, then prefailure algorithm ends. So, there is no need to move to the failed node’s location. If all distances are less than $R_c/2$, then DA asks its neighbors to move away from the DA to maintain the mandatory distance in all directions. If all distances are greater than the required distance, then DA asks its neighbors to move towards the DA to maintain such distance. There may a situation. When some nodes may be at distance equal to $R_c/2$, some nodes may be at distance less than $R_c/2$, and some have distance more than $R_c/2$ from DA, so, in such situation, DA broadcasts a single message having three parts: the first part of the message starts with “start,” then node IDs of those which have distances equal to $R_c/2$, and ends at “Eq.” Then, the second part of message starts with IDs of those nodes having distance less than $R_c/2$ and ends with “less” and the third part contains IDs of those nodes which have distance with DA more than $R_c/2$. After this, there is a footer of “more” and its ends with “end” which means end of a message. This message is broadcast by DA and all neighboring nodes read this message looking for their ID and responding as accordingly. If a node has distance equal to $R_c/2$, then it stays there and simply discards the message. If node falls in the category of having distance less than $R_c/2$, so it moves away from DA to make it $R_c/2$ and sends “done” message to DA. If distance from DA is greater than $R_c/2$, then that node calculates possible movement distance towards the distribution admin (DA) node by calculating overlap coverage area and how much it can move towards admin node by finding the least coverage area with its neighbors. After calculating possible to move distance, the node moves to that possible distance and sends “done” message to DA. When all the neighbors adjust their distances, DA updates distance list and finds DM for node failure situation. However, there might be a chance of a node having more than one DA, so such node will find possible distance with the DA. It selects the DA with lesser possible distance with itself. There might be another situation that there are two or more admins in the neighborhood in this situation; none of DAs moves towards each other, as their other neighboring nodes require moving towards them. Detailed steps are explained in the proposed prefailure pseudocode in Algorithm 1.

3.4.2. The Proposed Postfailure Algorithm. Mobile nodes in a network become uniformly distributed in AOI with the help of prefailure algorithm, to guarantee connectivity and coverage. There might be some nodes which cannot maintain $R_c/2$ with their neighbors, so recovery nodes are assigned to them. Now, for such nodes, which cannot maintain $R_c/2$, a problem occurs when some nodes fail in the network creating connectivity and coverage hole behind. To resolve this issue, a proposed node failure algorithm is presented in this paper. When a node finds one of its neighboring nodes has failed, for example, consider a scenario given in Figure 4. Suppose that node $F$ finds node $F$ which has failed. Node $A$ will have the status of failed node $F$ in its list that either $F$ is a cut-vertex (CV), intermediate node (IN), or leaf node (LN). So, node A will react to this situation according to the status of failed node $F$. If failed node $F$ is a cut-vertex, then it has at least four recovery nodes. If it has four or two recovery nodes, then all of them move their possible movement distance to restore connectivity. If they still not find each other, then they will increase their communication range and send message to sink node for redundant node as there is no other possibility to restore both coverage and connectivity. If there is only one node for as recovery node for CV, then it will move its possible distance to restore connectivity; if recovery nodes still do not find any node to restore connectivity, then they increase their range and send message to sink node for redundant node. If failed node $F$ is IN or LN, then the same procedure will be adopted; for IN, two recovery nodes or only one recovery node will move their possible distance or increase their range. For LN, only one recovery node will follow the above procedure. Pseudocode of this is given in Algorithm 2.

3.5. Energy Model. An energy model which we have used in this paper is illustrated in [25]. Equations (12) and (13) show the relationship of required energy for transmitting and receiving a B-bit data packet. Energy per bit consumption of receiver circuitry is given by (13). For more details, the readers are recommended to read details in [25].
Input: AOI and Rc

(1) Begin
(2) Find \( (N_i)_{i=1,2,3,\ldots} \) number of neighbors
(3) Check for PC = physical constraint
(4) If PC = true
(5) Send (owned ID, CM) //CM = cannot move
(6) Go to (step 48)
(7) End if
(8) If receive (some IDs, CM)
(9) Update (some IDs, CM) //in neighbor’s list
(10) End if
(11) If PC = false
(12) Compare (owned ID with \( N_i' \) IDs)
(13) End if
(14) If owned ID > all \( N_i' \) IDs
(15) Send (owned ID, distribution admin) to rest of neighbors
(16) Calculate (distance with each neighbor)
(17) If all neighbors’ distances = \( R_c/2 \)
(18) End if
(19) If distance < \( R_c/2 \)
(20) Send (move away) //to maintain \( R_c/2 \) distance
(21) End if
(22) End if
(23) Else
(24) Send (calculate possible movement distance) //to those neighbors whose distance is >\( R_c/2 \)
(25) Wait
(26) Receive (possible movement distances of all neighboring nodes)
(27) Calculate (actual distance—possible distance to move) //to find out which node can maintain \( R_c/2 \)
(28) Send (adjust, distance to move)
(29) End if
(30) If receive (move away)
(31) Move //to maintain \( R_c/2 \)
(32) Else if receive (calculate possible movement distance)
(33) If distribution admin > 1
(34) Send (CM) //to distribution admins
(35) Else
(36) Calculate (possible movement distance) //keeping in mind \( d \leq 2 \times R_c \) and by averaging possible movement distance of each neighboring node
(37) Send (adjustable distance) //to distribution admin
(38) End if
(39) End
(40) Else if receive (adjust, distance to move)
(41) Move //to adjust that \( R_c/2 \)
(42) End if
(43) Update all neighboring node distances (di)
(44) If all di \# \( R_c/2 \)
(45) Go to (step 36)
(46) End if
(47) Else if all di = \( R_c/2 \)
(48) Determine directional matrix = DM
(49) End if
(50) End
(51) Determine DM
(52) If DM = LN //leaf node
(53) Go to (step 66)
(54) Else if DM = IN //intermediate node
(55) Go to (step 64)
(56) Else if DM = CV //CV = cut-vertex
(57) Check availability of four recovery nodes
(58) If availability = false
(59) Go to (step 64)

Algorithm 1: Continued.
If availability = true
Set at least four neighboring nodes as recovery nodes
End if
End

Set at least two neighboring nodes as recovery nodes
If availability of two nodes = false
Set at least one neighboring node as recovery node
End if
End

ALGORITHM 1: Prefailure algorithm.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>If (node A detects, node F has failed)</td>
</tr>
<tr>
<td>2</td>
<td>If distance = ( R_c / 2 )</td>
</tr>
<tr>
<td>3</td>
<td>Find (new neighbors)</td>
</tr>
<tr>
<td>4</td>
<td>If (new node found)</td>
</tr>
<tr>
<td>5</td>
<td>Restore (connectivity)</td>
</tr>
<tr>
<td>6</td>
<td>Else if (new node not found)</td>
</tr>
<tr>
<td>7</td>
<td>Move (possible movement distance)</td>
</tr>
<tr>
<td>8</td>
<td>Else if (new node not found)</td>
</tr>
<tr>
<td>9</td>
<td>Go to (step 39)</td>
</tr>
<tr>
<td>10</td>
<td>Restore (connectivity)</td>
</tr>
<tr>
<td>11</td>
<td>Else if distance ≠ ( R_c / 2 )</td>
</tr>
<tr>
<td>12</td>
<td>Go to (step 18)</td>
</tr>
<tr>
<td>13</td>
<td>End if</td>
</tr>
<tr>
<td>14</td>
<td>End if</td>
</tr>
<tr>
<td>15</td>
<td>End if</td>
</tr>
<tr>
<td>16</td>
<td>End</td>
</tr>
<tr>
<td>17</td>
<td>Else</td>
</tr>
<tr>
<td>18</td>
<td>If DM = CV</td>
</tr>
<tr>
<td>19</td>
<td>Move (towards node F by possible movement distance)</td>
</tr>
<tr>
<td>20</td>
<td>Find (other recovery nodes)</td>
</tr>
<tr>
<td>21</td>
<td>If new node found</td>
</tr>
<tr>
<td>22</td>
<td>Restore (connectivity)</td>
</tr>
<tr>
<td>23</td>
<td>Else if (new node not found)</td>
</tr>
<tr>
<td>24</td>
<td>Go to (step 39)</td>
</tr>
<tr>
<td>25</td>
<td>End if</td>
</tr>
<tr>
<td>26</td>
<td>Else if DM = IN</td>
</tr>
<tr>
<td>27</td>
<td>Move (towards node F by possible movement distance)</td>
</tr>
<tr>
<td>28</td>
<td>Find (other recovery nodes)</td>
</tr>
<tr>
<td>29</td>
<td>If new node found</td>
</tr>
<tr>
<td>30</td>
<td>Restore (connectivity)</td>
</tr>
<tr>
<td>31</td>
<td>Else if new node not found</td>
</tr>
<tr>
<td>32</td>
<td>Go to (step 39)</td>
</tr>
<tr>
<td>33</td>
<td>End if</td>
</tr>
<tr>
<td>34</td>
<td>Else if DM = LF</td>
</tr>
<tr>
<td>35</td>
<td>Move (towards node F by possible movement distance)</td>
</tr>
<tr>
<td>36</td>
<td>Find (new node)</td>
</tr>
<tr>
<td>37</td>
<td>Restore (connectivity)</td>
</tr>
<tr>
<td>38</td>
<td>If new node not found</td>
</tr>
<tr>
<td>39</td>
<td>Increase ( R_c )</td>
</tr>
<tr>
<td>40</td>
<td>Send (redundant node) // to sink node</td>
</tr>
<tr>
<td>41</td>
<td>End if</td>
</tr>
<tr>
<td>42</td>
<td>End</td>
</tr>
</tbody>
</table>

ALGORITHM 2: Postfailure algorithm.
\[ E_{Tx}(B, d) = \begin{cases} (E_{elec} + \varepsilon p d^2)B, & d < d_0, \\ (E_{elec} + \varepsilon mp d^2)B, & d \geq d_0, \end{cases} \]
\[ E_{Rx}(B) = E_{Rx-elec}B, \]
\[ E_{ranged}(n) = E_{max} - E_{Tx}(B, d) - E_{Rx}(B). \]
In Figure 8, the average number of packets exchanged during the procedure of connectivity restoration is shown. UDRA results in the minimum number of packets exchanged, while the maximum number of packets is exchanged by GSR. The major reason behind this is that, in UDRA, the emphasis is on minimizing the number of neighbors involved during the recovery process, which results in reduction of cascaded movement of nodes, hence reducing the total number of messages exchanged. On the other hand, due to excessive movement of nodes, GSR results in the maximum number of packets exchanged. So, it can be concluded that UDRA is more energy efficient as compared to AUR and GSR algorithms. Another important feature is the scalability associated with the reduction of the transmission/receiving of messages. Due to least number of messages exchanged by UDRA, it proves to be more scalable as compared to AUR and GSR protocols. In sensor networks, connectivity restoration is of immense importance and a technique must be capable of restoring the connectivity. During this work, our major emphasis was to design a
prefailure and postfailure connectivity restoration protocol capable of achieving effective connectivity restoration by relying on minimal node mobility, minimal reduction in field coverage, and consuming minimal energy of nodes by reducing the overall number of exchanged messages. The solution proposed by UDRA achieves all the above goals. The performance of the UDRA was evaluated by using extensive simulations and by comparing its performance with AUR and a GSR technique; its effectiveness is proven.

5. Conclusions and Future Work

A novel algorithm “Uniform Distribution and Recovery Algorithm (UDRA)” is proposed in this paper. This algorithm provides both connectivity and coverage restoration solution for wireless sensor networks in an integrated manner. In Table 5, it is clear that average distance travelled by the sensor nodes is only 3000 meters. This distance is very less than 13000 meters and 15000 meters. So, having less average distance travelled proves that the proposed algorithm is an energy efficient solution. Most of the energy stored in batteries is consumed by travelling. If an algorithm is not energy efficient, then batteries of nodes will be drained off with the passage of time. This makes the problem worst and decreases the network lifetime. Furthermore, these results show that the number of messages exchanged by the proposed algorithm is very less as compared to RIR and AUR. This decreases the message overhead in the network. Greater message overhead causes delay in recovery process as it takes a lot of time by communication messages. Another finding from Table 5 is that in the proposed technique only 70 nodes are relocated; this gives an upper hand to the proposed algorithm over RIR and AUR as more relocation of nodes needs more energy used. The other advantage of a smaller number of relocated nodes is that relocation of nodes can cause coverage loss. So, the lesser relocated nodes are, the lesser the coverage loss will be. The last most important finding of Table 5 is the percentage reduction in the field coverage. In UDRA, the percentage in field coverage reduction is only 3% after recovery process. This shows that UDRA outperforms its competitor baseline algorithm. Less percentage in field coverage reduction ensures that most of the network or AOI is monitored and less information from AOI is lost. The cost of these advantages would be that increased number of nodes is required to cover certain AOI, which is still affordable for mission critical applications. The detailed analysis would be done in the future.

Data Availability

No data were used to support this study. We have conducted the simulations to evaluate the performance of the proposed protocol. However, any query about the research conducted in this paper is highly appreciated and can be asked from the principal authors (Rab Nawaz Jadoon (rabnawaz@mail.ustc.edu.cn) and Muhammad Amir Khan (amirkhan@cuiatd.edu.pk)) upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgments

Dr. Rab Nawaz Jadoon personally thanks COMSATS University, Islamabad, Abbottabad Campus, and School of Information Science and technology, USTC Hefei, China, that extended their full support by providing all key resources during the implementation and all afterward phases of this project. Also, special thanks are due to Prof. WuYang Zhou, because of his continuous encouragement and massive support academically, financially, and socially during this research work. This work was financially supported by the National Natural Science Foundation of China (Grant no. 61631018).

References


