Connectivity Restoration in Wireless Sensor Networks using Steiner Tree Approximations

Mohamed Younis and Rahul Waknis
Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County
Baltimore, Maryland, USA
younis, rahulw1@umbc.edu

Abstract—Wireless sensor nodes are symbiotic when deployed in an activity region and heavily rely on each other for successful transmission of data. Therefore, failure of some nodes can possibly partition the network. Since these networks often operate unattended, nodes need to collectively maintain connectivity and resolve any reachability problem. Most published approaches for restoring connectivity are based on a single underlying principle of replacing the failed node without considering the possible fact that the location of the failed node could have been the reason for its failure. These approaches also tend to trigger a cascaded relocation of many nodes resulting in increased overhead. This paper presents a novel solution that pursues rearrangement of nodes while limiting the scope of the recovery to the vicinity of the failed node. The connectivity restoration is modeled as a variant of the Steiner tree formation problem and solved using novel heuristics. The proposed approach is validated through simulation.

I. INTRODUCTION

With an ever increasing range of applications, Wireless Sensor Networks (WSNs) have grabbed the attention of researchers and practitioners [1]. WSNs can serve numerous domains. They can be deployed along national borders to detect infiltrations and drug and human trafficking, in remote mines to test for gases, on distant planets to check for sign of life, and in forests to detect fires. These application environments force the network to operate in harsh conditions and without direct human control. Upon deployment, sensor nodes are to form a connected network topology and operate autonomously and collaboratively to achieve a mission.

Sensor nodes may fail due to exhausting their battery, hardware faults, externally inflicted damage caused by the inhospitable environment, etc. Given the collaborative node operation, a strongly connected network topology would be required at all times. A node failure would negatively affect the network connectivity and in the worst case, may partition the network into mutually unreachable blocks. Given the unmanned nature of its operation, the network cannot rely on the deployment of a substitute for the failed node and would thus need to self-configure to restore connectivity.

A mobile sensor network is a class of WSN where nodes have the capability to move. With this capability, nodes can reposition themselves for the purpose of better coverage, saving energy and also for maintaining connectivity in case of failures [2][3]. Node repositioning has been shown to be an effective method for restoring connectivity in a network in case of node failures [4][5]. However these approaches rely on a single underlying principle of “trying to replace the dead node” without taking into consideration the fact that the location of the failed node might not have been optimal and which could have been the reason for its failure. These approaches also tend to trigger a cascaded movement of nearby nodes resulting in increased overhead and widening the scope of the recovery throughout the network.

This paper pursues a novel approach for recovering from a loss of connectivity caused by a node failure. The idea is to rearrange the network topology in the vicinity of the failed node. Basically, the 1-hop neighbors of the dead node are repositioned to ensure strong network connectivity. However, unlike most prior work, the location of the failed node is avoided when re-stationing the other nodes. The connectivity restoration is modeled as a special type of Steiner tree formation problems, in which the number of Steiner points is constrained. Since Steiner tree formation is an NP-Hard problem, heuristics are pursued. In addition to restoring connectivity, the proposed approach localizes the scope of the recovery. The performance of the recovery algorithm is validated through simulation and is shown to surpass contemporary schemes in the literature.

The next section describes the assumed system model and defines the problem; Section III gives an overview of related work; Section IV explains our approach in detail; Section V describes the validation experiments and analyzes the simulation results; and finally Section VI concludes the paper.

II. SYSTEM MODEL AND PROBLEM STATEMENT

The paper deals with Connectivity Restoration after a node failure through Rearrangement (CRR) of nodes in the vicinity. The approach applies to a network of mobile sensor nodes. Even though this approach can be extended to include a hierarchical topology consisting of a mixture of stationary and mobile nodes, we will be assuming that all the nodes are mobile. Sensor nodes can reach each other via multi hop routes. Intermediate nodes which forward data to other reachable nodes are called relays. This paper does not distinguish between relays and sensor nodes and assumes that every node has the capability to relay packets to its neighbors. We assume that such a network has already been formed by using any of the strategies mentioned in [2] and is fully functional. Individual nodes in such a network communicate with their neighbors which are determined by the node’s radio range. For easing the presentation, we assume that all nodes in the network have identical communication range.

The multi-hop nature of the network gives rise to dependencies among nodes. A failure of one node can affect other nodes or worse, the entire network. Various reasons for
The failure of \( S_5 \) will partition this network into 4 disjoint blocks. Failure have been mentioned in Section I. In Graph Theory, such critical nodes are called cut-vertices. Node \( S_5 \) in the Figure 1 is considered a cut-vertex.

The CRR algorithm views the recovery from partition problem as a node placement problem and rearranges certain nodes to achieve three main objectives; (1) recover from a network partitioning, (2) limit the scope of the recovery to the vicinity of the failed node, and (3) reduce the travel overhead imposed on the participating nodes. It is assumed that during the recovery process, there are no additional node failures. We also assume that every node has 3-hop neighbor information, including the position coordinates. This paper also deals with the recovery at the network layer, and assumes that neighbor discovery, link creation, maintenance and termination are handled by lower layers of the communication protocol stack.

III. RELATED WORK

To tolerate occasional node failures, two categories of approaches have been proposed in the literature. The first category pursues a provisioned methodology by deploying redundant nodes so that the network operation continues without interruption. For example, the approach of [7][8] is to establish 2-connected topology in order to sustain network connectivity after a single node failure. In the second category nodes are repositioned to restore lost connectivity and/or coverage [3][4][5][9]. It is worth noting that node repositioning has also been pursued for enhancing various performance metrics in sensor networks [2].

Published connectivity-restoration approaches can be classified according to whether the impact of the failure is analyzed before the recovery process is initiated. For example, RIM [9] does not check whether the failed node represent a cut-vertex or not. The main philosophy is that the analysis requires messaging overhead. Therefore, RIM only requires a node to be aware of its directly reachable neighbors and shrinks the network inward to make these neighbors directly reachable to each other. Any impact on the further nodes, i.e., loss of a link, will be tolerated by cascaded relocation towards the dead node. Obviously this approach may be expensive since it may overreact and mobilize the entire network for the failure of a non-critical node. In addition, unlike the proposed CRR approach, RIM often engages all nodes in the network and this widen the scope of the recovery.

The other class of work reacts only if the failed node is a cut-vertex. Examples include DARA [4] and PADRA [5]. DARA recovers from a cut-vertex failure by replacing it with one of its neighbors. A neighbor node with the least degree is considered as a good candidate for relocation. Any impact due to its movement is handled by cascaded relocations of nearby nodes. PADRA classifies the nodes into Dominator and Dominatee. The failure of the former triggers a recovery which is pre-planned with the closest healthy dominatee taking its place. The failure of a dominatee is ignored.

As mentioned the above approaches try to maintain neighbor links by cascading the movement of a node until the all links to neighbors remain intact or until the cascading propagates to the leaves of the network. Hence as the radius of the network increases, the cascaded movement becomes quite a costly affair. If the reason for the failure of the node is its location, then implementing approaches like DARA and PADRA which blindly follow the dead node can have serious consequences and lead to the entire network been taken down.

IV. RECOVERY ALGORITHM

A. Problem Analysis

The CRR approach is based on a single intuitive principle, the location of the dead node which caused the failure cannot drive the entire recovery process of the network. In other words, unlike the previous approaches, the CRR does not make any assumptions about the optimality of the location of the dead node. Instead, it opts to rearrange the network topology in the vicinity of the dead node in order to restore connectivity and avoid the involvement of further nodes. Thus, the recovery problem is modeled as a node positioning problem to connect some terminals. The goal of the repositioning is to limit the number of involved nodes, which makes the recovery a Steiner Tree formation problem.

Generally forming a Steiner tree is an optimization problem to minimize the overall cost of establishing a spanning tree of terminals by introducing a set of vertices called Steiner points. Steiner points are found by locating the center of a disc of radius \( x \) which spans some terminals (at the most 3) in the spanning tree. When such a point is located, removing the edges that link these terminals in the spanning tree and connecting them to the center of the disc yields a lower cost spanning tree as shown in Figure 2. The \((SP_1, S_5, S_6, S_7)\)-like structure is referred to hereafter as 3-stars.

Figure 1: Part of a Sensor Network showing a cut-vertex node \( S_5 \).
CRR does not rely of the availability of spares and utilizes only existing nodes. In other words, the number of Steiner points is constrained. This makes the recovery problem a unique variant of the Steiner tree formation problem. In fact, the closest problem formulation is the bottleneck Steiner problem [10], which imposes a bound on the Steiner points count. However, the problem considered by CRR is different in the sense that the link length cannot exceed the communication range of a node “R”. Therefore, none of the existing heuristic solutions for general and bottleneck Steiner tree problem would directly apply.

B. The CRR Recovery Procedure

Two main objectives are targeted by the design of CRR; (1) the position of the failed node is to be avoided, and (2) the recovery overhead in terms of travel distance is to be reduced. The following describes the major steps of CRR:

Determining the scope of the recovery: The recovery process begins when the 1-hop neighbors of \( S_d \) miss heartbeat messages from \( S_d \), and each of them independently arrives at the conclusion of the failure of \( S_d \). These direct neighbors can then check if all 2-hop neighbors of \( S_d \) can still be reached. In other words, surrounding nodes check whether the dead node was a cut-vertex. CRR employs the approach of [6] for a distributed assessment on whether the failed node is a cut-vertex and invokes the recovery procedure only if it is so. The set of 1-hop and 2-hop neighbors of \( S_d \) will be referred to hereafter as \( \text{SN}_1 \)-hop and \( \text{SN}_2 \)-hop respectively.

The CRR approach tries to limit the scope of the recovery to the vicinity of the dead node. Basically, the connectivity restoration process will take place within the polygon bounded by the 2-hop neighbors of \( S_d \). The rationale is that forming a strongly connected sub-graph of these nodes will ensure the network is recovered. The idea is to place 3-stars in such a polygon in order to form a Steiner tree while considering the 2-hop neighbors of \( S_d \) as terminals. The entire recovery process of CRR involves forming 3-stars in the neighborhood of \( S_d \) by moving nodes from \( \text{SN}_1 \)-hop, and some from \( \text{SN}_2 \)-hop if needed.

Finding new locations for the 1-hop nodes: After identifying the participating nodes and boundary of the recovery polygon, the next step is to explore options for rearranging the nodes in \( \text{SN}_1 \)-hop so that the connectivity is restored. First every pair of adjacent terminals \( T_p \) and \( T_q \), i.e., two nodes in \( \text{SN}_2 \)-hop, is considered with \( S_d \), and a 3-star is formed by introducing a Steiner point \( \text{SP}_r \). Adjacency in this context means that the terminals are next to each on the line defining the boundary of the polygon that has all 2-hop neighbors as vertices. Thus, two adjacent terminals are not necessarily connected. To find \( \text{SP}_r \), the \( k \)-restricted loss-contracting algorithm (k-LCA) [11] is used. The k-LCA is the best known approximation algorithm to solve Steiner tree problems, where \( k \) denotes the number of terminals. In case of CRR, \( k \) is set to 3 in order to factor in the node’s communication range. Basically, CRR strives to maximize the number of terminals reached by the individual Steiner points. Therefore, if \( \text{SP}_r \) is less than \( R \) units away from \( T_p \) and \( T_q \), these two terminals are considered covered. Otherwise, \( \text{SP}_r \) is just ignored. If none of the identified \( \text{SP}_r \)'s could cover two terminals, a Steiner point is identified to cover just one terminal in addition to \( S_d \) (simply the midpoint between the terminal and \( S_d \)). The set of identified Steiner points are further pruned by checking for redundantly covered terminals. If \( T_p \) is covered by \( \text{SP}_x \) and \( \text{SP}_y \), \( T_q \) is covered by \( \text{SP}_y \) and \( \text{SP}_z \), and \( T_r \) and \( T_s \) are adjacent terminals, then \( \text{SP}_y \) is considered redundant and gets eliminated.

Next, all Steiner points that are included and uncovered terminals are considered as vertices and a new polygon is formed. Every three of these vertices \( V_r, V_s \) and \( V_t \) that are adjacent on the newly-formed polygon, are considered as terminals and the k-LCA algorithm is applied to form a 3-star. Again a newly identified Steiner point that lies more than \( R \) units away from more than one vertex among \( V_r, V_s \) and \( V_t \), is ignored unless none of the Steiner points for all groups of 3 vertices qualify for inclusion. In that case, one randomly picked Steiner point is picked. The vertices that are reachable to a committed Steiner point is marked as covered and are not considered any further. The set of newly identified Steiner points are also pruned as explained above. This process is repeated until all vertices (nodes in \( \text{SN}_2 \)-hop and added Steiner points) are covered. Note that the dead node \( S_d \) is considered as a terminal in the beginning in order to prevent considering its position as a candidate Steiner point.

Node rearrangement for restoring connectivity: The identified Steiner points need to be populated for \( \text{SN}_2 \)-hop to become connected. For that, CRR utilizes \( \text{SN}_1 \)-hop by moving each of them to a Steiner point. Obviously, one cannot guarantee that \( |\text{NSP}| \) equals \( |\text{SN}_1 \)-hop|). The following three scenarios arise:

i. \( |\text{NSP}| = |\text{SN}_1 \)-hop|): This is considered the ideal scenario. Every Steiner point will be occupied by the nearest 1-hop neighbor of \( S_d \). In other words, each of the nodes in \( \text{SN}_1 \)-hop reposition to one of the Steiner points.

ii. \( |\text{NSP}| < |\text{SN}_1 \)-hop|): This will be handled like the scenario above except some neighbors of \( S_d \) will not need to move.

iii. \( |\text{NSP}| > |\text{SN}_1 \)-hop|): This scenario will need the engagement of \( \text{SN}_2 \)-hop nodes. Basically, every member of \( \text{SN}_1 \)-hop will move to its nearest Steiner point that lies inward towards \( S_d \). If there is not such a point, a node in \( \text{SN}_1 \)-hop will simply move to the nearest Steiner point. The remaining Steiner points will be occupied by the closest \( \text{SN}_2 \)-hop node. The departure of a \( \text{SN}_2 \)-hop node may need cascaded relocation to ensure that connectivity is not lost in another part of the network. For cascaded relocation, CRR employs DARA [4], which picks a replacement of the moved node based on node degree and proximity.

Figure 3 illustrates how the CRR algorithm restores the connectivity in the network of Figure 1 after node \( S_d \) fails.

V. PERFORMANCE VALIDATION

A. Simulation Environment and Performance Metrics

The CRR approach is validated through simulation. In the experiments, we create a WSN of up to 200 nodes that are initially placed at random in an area of 1000m x 1000m. The following metrics are pursued to assess the performance:
- **Total traveled distance**: reports the total distance that the involved nodes collectively move during the recovery. This can be envisioned as a network-wide efficiency assessment.

- **Number of relocated nodes**: reports how many nodes move during the recovery. This metric assesses the scope of the connectivity restoration process within the network.

- **Area coverage**: Unless the node density is really high, the loss of a node negatively affects the network coverage. This metric tracks how the CRR spread the nodes in the area to increase coverage while restoring connectivity.

The following parameters are used to vary the characteristics of the WSN topology in the different simulation experiments and study the implications on the performance of CRR:

- **Number of Deployed Nodes (N)**: This parameter affects the node density and the WSN connectivity.

- **Communication range (r)**: All nodes in the experiments have the same communication range r. The value of r affects the connectivity of the initial WSN topology.

The CRR algorithm is compared to the Recovery through Inward Motion (RIM) approach [9]. RIM also avoids placing a substitute at the position of $S_p$. However, RIM moves the nodes towards $S_p$ in order to reconnect the network.

### B. Simulation Results

We have simulated WSN topologies with some combinations of values of r and N. The N is selected from a set of {60, 80, 100, 150, 200} while r is varied in the range [50, 200]. When changing r, the network size is fixed at 100 nodes, and r is set to 100 meters while varying N. For every topology, CRR and RIM are run after failing a randomly selected node. The results of the individual experiments are averaged over 30 runs. All results are subjected to 90% confidence interval analysis and stays within 10% of the sample mean.

**Total travelled distance**: Figure 4-(a) shows the distance that the nodes collectively traveled to restore connectivity. As indicated in the graph, CRR always outperforms RIM. Furthermore it is clear that CRR is able to bear the load of increased node density. In fact, CRR performs better in dense topologies in contrast to RIM since the average node degree grows, i.e., more 1-hop neighbors for the failed node, and thus fewer cascaded relocation would be needed. RIM on the other hand proves to be costly in dense networks since many nodes get engaged in cascaded relocation. Figure 4-(b) shows the effect of radio range on the total travel distance. The network wide cascading of RIM produces an exponential rise in the traveled distance since the connectivity of the network is higher and many nodes had to move. Meanwhile, the increase of r boosts the node degree and enables CRR to limit the scope of the recovery to vicinity of the failed node.

**Number of relocated Nodes**: Figure 5-(a) reports the number of nodes that moved to tolerate the node failure as the network size increases. The figure clearly indicates that CRR manages to engage only 3-5% of the nodes and avoids cascading in dense networks due to the presence of adequate neighbors. CRR is hence successful in limiting the scope of recovery.

---

**Figure 3**: Detailed explanation of how CRR restores connectivity for the example in Figure 1; (a) the sets $SN_{1-hop}$ and $SN_{2-hop}$ are marked. (b) Steiner points are picked considering every pair or adjacent 2-hop neighbors and the dead node. (c) Steiner points that are not reachable to two 2-hop nodes are ignored. Redundant Steiner points are eliminated. (d) Vertices, i.e., uncovered 2-hop nodes and Steiner points, are identified and k-LCA algorithm is applied for every 3 adjacent vertices. (e) Vertices that are not reachable to two vertices are ignored. Redundant Steiner points are eliminated. All 2-hop neighbors are covered. (f) 1-hop nodes moves to the nearest Steiner points in the direction of the failed node or simply the closest Steiner point otherwise.
RIM, on the other hand, engages many nodes in the relocation process. As shown in Figure 5-(b), long communication ranges is a disadvantage for RIM since the network connectivity grows. CRR as mentioned above benefits from the increased node degree to reduce the scope of the recovery.

**Area coverage after relocation:** Figure 6 shows the coverage of the network after the relocation process ends as the number of nodes and communication range increase. The coverage range of a node is set to 50 meters in these experiments. Both RIM and CRR maintain coverage up to a point where the side effect of the RIM mechanism starts to surface. This observation, which holds when changing either $r$ or $N_h$, is attributed to the increased connectivity in the network. When the failed node has many neighbors RIM tends to unnecessarily move all of them very close to the failed node and shrink the network inward, leaving significant part of the outer area uncovered. CRR does a better job in dense networks as it avoids cascading. In addition, CRR tries to work with existing nodes to restore connectivity by placing them evenly in the area around the dead node decreasing redundant coverage.

**VI. Conclusion**

With the ever increasing use of wireless sensor networks (WSNs) in mission-critical applications serving unattended environments, it is essential that these networks autonomously recover from node failures. This paper has presented a novel approach for restoring severed connectivity after the loss of a node. Unlike most prior work, the proposed connectivity restoration through node rearrangement (CRR) algorithm opts to avoid the position of the failed node to minimize the risk of additional loss of resources and strives to localize the scope of the recovery. The CRR approach simply defines appropriate Steiner points to connect the 2-hop neighbors of the failed node and reposition the 1-hop neighbors at these points. Uncovered Steiner points are handled through cascaded relocation. The simulation results show that CRR is very effective for low as well as high density networks and significantly outperforms comparable published schemes.

**Acknowledgement:** This work is supported by the National Science Foundation, contracts # 0002270 and 0005844.

**References**