Mobility-based Internetworking of Disjoint Segments

Yatish K. Joshi and Mohamed Younis
Dept. of Computer Science and Electrical Engineering
University of Maryland Baltimore County
Baltimore, MD 21250
yjoshi1, younis@umbc.edu

Abstract—**Wireless sensor networks (WSNs) deployed in inhospitable environments are at increased risk of failure due to malicious enemy action, environmental causes or component malfunction. Large scale failure can divide a network into disjoint segments and disrupt its operation. Reestablishing inter-segment connectivity is of paramount importance to ensure that the network functions effectively. Mobile nodes within surviving disjoint segments can be used to serve as mobile data collectors (MDCs) to tour individual segments and reconnect them by ferrying messages. However finding shortest tours for MDCs is a NP hard problem. In this paper we present a mobility based polynomial time approach for internetworking of disjoint segments (MINDS) by using *k* MDCs to establish tours between segments. MINDs aims to not only minimize total tour length but also balance the load equitably between the *k* MDCs. MINDS uses a divide and conquer approach to successively split the larger tour into two around a central node at every step, until we obtain *k* optimized tours. MINDS is validated through simulation and is shown outperform competing schemes in the literature.**

Keywords: **Mobile data collectors, Fault recovery, Connectivity restoration, Fault-tolerance, Wireless sensor networks.**

I. Introduction

Usage of wireless sensor networks (WSNs) has become increasingly pervasive due to their versatility and low cost. Their ability to monitor and function in environments unsafe for humans has made them the solutions of choice for data gathering and surveillance applications in inhospitable setups like forests, deserts and battlefields. After deployment nodes are expected to coordinate with each other and form a network to carry out their designated tasks. Due to the harsh operating conditions or through enemy action like missile strikes or bombings in a battlefield, the network is susceptible to large scale damage that can partition it into multiple disjoint segments. These challenging setups do not allow for human intervention or replacement of failed components in a timely manner hence the network should be able to self-heal through an autonomous recovery process in order to prolong network lifetime.

Connectivity restoration strategies that depend on establishing a k-connected topology [1][2][3], or employing redundant nodes [4][5][6] are ineffective since the location of failure cannot be predetermined and provisioned for in advance. Therefore, recovery has to be reactive in nature. A number of reactive approaches are proposed in the literature to deal with the failure of individual nodes [7][8][9], however they do not scale to tolerate multiple node failures. AuR [10] recovers the network through inward motion towards the center of the deployment area although effective it requires all nodes to be mobile and cannot be applied if the network consists of a mixture of stationary and mobile nodes. The network partitioning problem can also be handled by deploying stationary relay nodes (RNs) to connect disjoint segments. The relay placement problem is equivalent to the Steiner Minimum Tree with Minimal Steiner Points and Bounded Edge-Length, which is an NP-Hard problem [11]. A number of heuristics [11][12][13][14] have been proposed in the literature for solving the SMT-MSPBEL problem. They all assume that a sufficient number of RNs are available for recovery, which may not be the case due to large scale damage or limited resources.

This paper opts to tackle the failure recovery problem when only a limited number of mobile nodes are available. A novel Mobility based approach for Internetworking of Disjoint Segments (MINDS) is proposed. MINDS utilizes the available *k* mobile nodes in the disjoint segments as MDCs to restore network connectivity. This solution is scalable as even one MDC is enough to provide an intermittent communication pathway between disjoint segments. MDCs travel on routes assigned to them and visit all the segments in their assigned tour collecting and disseminating data between the segments.

The design goal of MINDS is to utilize the k available MDCs to construct k balanced tours such that the total tour length is minimized and tour duties equitably distributed amongst the MDCs. To minimize tour length each segment is represented by a collection point that a MDC must come within transmission range of so that data exchange can take place. MINDS runs in rounds until all MDCs have been assigned. It constructs a Minimum Spanning tree (mst) using collection points as vertices and initially assigns one MDC to cover them all. It successively partitions the largest tour into 2 along the center until there are k tours. In each round the number of MDCs deployed is increased by one, this continues until all k MDCs have been employed.

The paper is organized as follows. The next section sets MINDS apart from related work in the literature. Section III defines the system model. Section IV describes MINDS in detail. Section V reports the simulation results. Finally, Section VI concludes the paper.

II. Related Work

A number of approaches have recently been proposed for recovering from large scale failure by deploying stationary RNs such as IO-DT [14], DarDs [15] and Dorms [16]. They all work under the assumption that the RNs required to restore connectivity are available in requisite quantities which may not be the case in a real world scenario. Meanwhile approaches like AuR [10], require all nodes to be mobile which would not be applicable across all deployment and application scenarios.

IDM-kMDC [17] and ToCS [18] pursue a solution based on touring segments using MDCs. ToCS cares about the maximum delay rather than travel overhead, and forms a star topology where MDCs do not necessarily rendezvous at the segments. Meanwhile, IDM-kMDC computes an *mst* over the collection points of all disjoint segments and then assigns an MDC to each *mst* edge. The tours are combined in a greedy manner in order to meet the MDC availability constraint. It runs in rounds and in each it selects two tours such that their merging cost, in terms of increased travelled distance, is the least amongst all possible combinations of two tours. In each round the number of required MDCs is reduced by one until the number of tours is equal to the number of available MDCs. Though effective the runtime complexity of IDM-kMDC is high, mainly due to the consideration of all tour combinations at each step.

Like IDM-kMDC, MINDS also constructs an *mst* over the collection points; however it assigns one MDC in each round until all *k* MDCs have been assigned. In the first round it assigns one MDC to cover all *mst* vertices in one tour. In the subsequent rounds, i.e., for 'k>1' it will iteratively break the largest tour by splitting the *mst* formed by the collections points that are part of the tour under consideration into two. Even splitting of the *mst* is done by finding its center. Thus, each round increases the number of employed MDCs by one. As proven later in the paper, MINDS converges faster than IDM-kMDC and can thus scale for large networks. Simulation results also confirm that MINDS yields more balanced tours while also minimizing the total tour length like IDM-kMDC.

III. System Model

The proposed MINDS approach considers a WSN composed of a mixture of stationary and mobile nodes that has been partitioned into disjoint segments due to failures. The stationary nodes are used for sensing and application specific tasks and communicate wirelessly with their neighbors within transmission range. We assume that all nodes have the same transmission range 'R'. The mobile nodes referred to as MDCs have the ability to move, a greater battery life and storage capacity. Although they can be used for sensing in an undamaged connected network we are concerned primarily in utilizing their mobility to reconnect a damaged network by ferrying data between the disjoint segments. Due to their versatility they are more expensive and thus their numbers limited, we assume that after a large scale failure we have k MDCs available to reconnect the network.

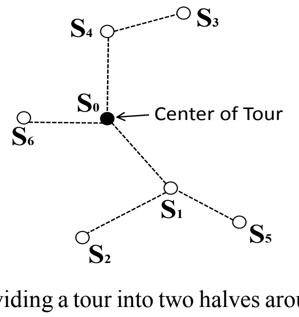


Figure 1: Dividing a tour into two halves around the center.

IV. Mobility based Internetworking

MINDS aims to reconnect 'n' disjoint segments using k MDCs. It first defines a collection point for each segment which acts as an interface for data pickup and delivery. Then, MINDS forms an *mst* over the set of collection points and forms a MDC tour for visiting these collection points to carry data among the segments. This tour is progressively split to finally define the segments that each of the individual MDCs will serve and the travel path for each MDC, as described below.

A. Determining the Center of the mst

If only one MDC is available, the *mst*-based tour is the final solution for MINDS and no further optimization is performed. To take advantage of the availability of multiple MDCs, the initial tour is split by associating a distinct subset of segments to individual MDCs. To do so, MINDS finds the center vertex on the *mst*. The center of a graph is the vertex for which the length of shortest path to the farthest vertex is the smallest. To find the center of a general graph, the shortest path between every pair of vertices is found using Floyd-Warshall algorithm which has $O(n^3)$ runtime complexity. However, MINDS only considers the *mst* edge and thus a depth first search suffices.

Theorem 1: The center vertex of an *mst* can be determined in $(n \log n)$.

Proof: Each collection point S_i can concurrently conduct a Depth First Search (DFS) to find the distance to the farthest vertex. The time complexity of which is $O(E)$ or $O(n-1)$ as there are *n-1* mst edges over *n* vertices. The center is the vertex whose distance to the farthest vertex is the smallest and can be determined by sorting the 'n' values from each S_i in $O(n \log n)$, which dominates the $O(n)$ time required for DFS.

B. Tour Splitting

MINDS successively selects the longest assigned tour to split if a MDC is still available. It determines the *mst* center of the collection points in the tour around which the split will occur. This is easy to do when the node degree of the center is two, by making a tour for each of the two partitions of the *mst* that are connected at the center and making the center the MDC rendezvous point for these two tours. If the node degree at the center is more than two, the partitioning is slightly more complex. Figure 1 depicts a *mst* comprising of S_0-S_6 which is to be partitioned. The center of the *mst* S_0 has a node degree of 3. We view the center in this case to be the root of a tree and its 3 neighbors being children. The first group is formed out of the branch that leads to the farthest collection point from the center, which has already been found during the center identification procedure. In this case S_2 is the farthest from S_0 and the first group consists of S_0- S_1- S_2- S_5. The second group consists of collection points on the branch that contains the furthest point from the center that is not part of the first group, i.e., S_0-S_4-S_3 for the example in Figure 1. The remaining branches, like S_6 join the group that has the closest 2-hop neighbors, meaning the distance between the child of the center on that branch and the other children of the center belonging to the two groups, will be the deciding factor. In Figure 1, S_6 is added to the second group as it is closer to S_4 than S_1. This results in splitting the *mst* into S_0- S_1- S_2- S_5 and S_0-S_4-S_3-S_6. The time complexity of splitting is at most $O(n)$.

C. Forming MDC Tours

Once partitioned, the MDC travel path must be determined. A tour T_s is defined as the shortest possible cycle that visits every collection point in its partition and returns to its initial position. A MDC must pass within the transmission range of a collection point so that data exchange can take place. We adopt the tour formation procedure of [17], which depends on the number of segments in a tour, as follows:

No of Segments =2: This is considered the simplest case where a MDC serves on the edge between two segments, as shown in Figure 2(a). The shortest path between two points is a straight line connecting them. If the distance between two points is $R < d \leq 2R$, where R is the communication range of a node, the MDC will not move and stay at the midpoint of the line connecting the points. If $d > 2R$, the tour stops are located at the intersection of the transmission discs of S_1 and S_2 of radius R and the line connecting them. The MDC will move back and forth between the tour stops to enable S_1 and S_2 to exchange data.

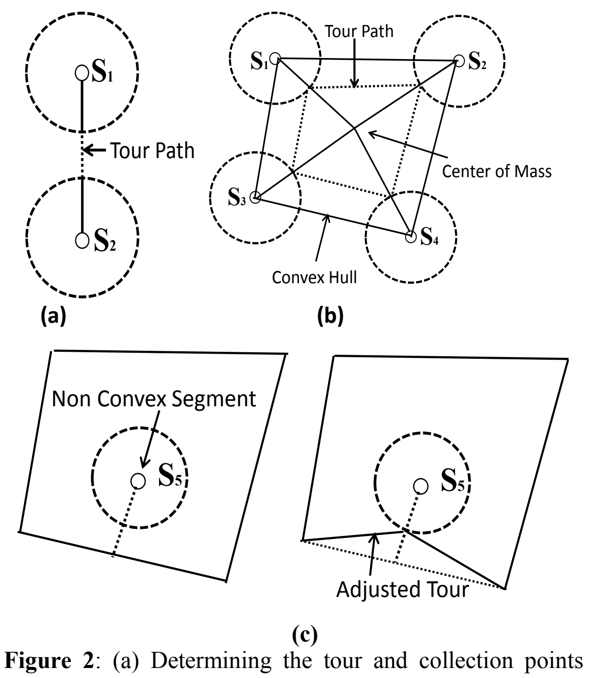


Figure 2: (a) Determining the tour and collection points between an mst edge, (b) MDC tour calculation for more than 2 segment collection points. (c) Accounting for non-convex hull segment collection points in a tour path.

No of Segments >2: In case a tour has more than two segments, MINDS constructs a convex polygon using Graham Scan Algorithm and determines the center of mass. The point of intersection of the transmission disc of a collection point with a line connecting it to the center of mass serves as a tour stop for the MDC as shown in Figure 2(b). Tour stop for a non-convex point is computed by finding the point of intersection of the perpendicular line from the collection point to the closest tour edge and its transmission disc as shown in Figure 2(c).

D. Stopping Criteria

Once a tour has been partitioned into two smaller tours, the number of tours is compared to the MDC count. If some MDCs are not yet employed, the longest tour will be split following the steps described above, and so on until k tours are established.

E. Illustrative Example

Figure 3 shows the application of the MINDS on a network that has been partitioned into 6 disjoint segments S_0-S_6 and has 3 MDCs; each segment is represented by a single collection point. MINDS first constructs a *mst* to connect all collection points together as shown in Figure 3(a). Since the number of available MDCs is greater than 1, the center of the *mst* is found and it is split into 2 tours S_0-S_4-S_3 and S_0-S_2-S_1-S_5 around center S_0 as seen in Figure 3(b). In each round the longest tour is split around its center, until all available MDCs have been employed. In Figure 3(b), after defining the 2 tours, tour S_0-S_2-S_1-S_5 is the longest and is split further around S_1 into S_1-S_0-S_2 and S_1-S_5. At the end in Figure 3(c), all 3 MDCs have been utilized terminating MINDS.

Figure 3(d) shows the topology constructed by the competing IDM-kMDC approach. IDM-kMDC initially assigns a MDC to each *mst* edge resulting in 5 (i.e., n-1) tours and in each round identifies 2 MDC tours that can be merged for the least cost, hence in each round the number of MDCs deployed is pruned by one until the number of tours equals the available MDCs. Comparing Figures 3(c) and 3(d) it is clear that MINDS produces more balanced tours and in this case results in a smaller total tour length. IDM-kMDC is a greedy approach and in each step needs to compare all two tours combinations to find a pair with the least merging cost, a process that has high runtime complexity for large networks.

Theorem 2: The runtime complexity of the MINDS algorithm is $O(kn \log n)$.

Proof: Most of the work in MINDS is done in the while loop which runs at the most $(k-1)$ times until all k MDCs are employed. Within the loop we know from Theorem 1 that the center of the *mst* can be determined in $O(n \log n)$, Splitting a tour takes at most $O(n)$ where each segment is considered and added to the one of two partitions. The cost of determining the optimized tour can be bounded by finding the convex hull through Graham scan algorithm, whose

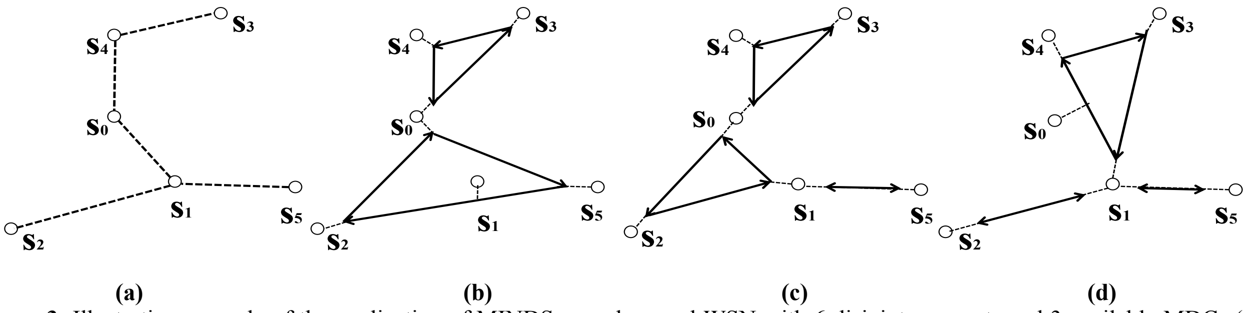


Figure 3: Illustrative example of the application of MINDS on a damaged WSN with 6 disjoint segments and 3 available MDCs (a) Minimum spanning tree connecting the disjoint network segments, (b) First division of *mst* around the center S_0, (c) MDCs tours formed by MINDS, (d) Tours formed by the competing IDM-kMDC approach.

runtime complexity is $O(n \log n)$ Therefore the runtime complexity of an iteration is $O(n \log n) + O(n) + O(n \log n)$, which is $O(n \log n)$. Since the while loop iterates $(k-1)$ times, the runtime complexity for MINDS is $O((k-1)n \log n)$ or $O(kn \log n)$. Since $k < n$ we can also say that MINDS is $O(n^2 \log n)$. It is worth noting the runtime complexity of IDM-kMDC is $O(n^4 \log n)$ [17], which is significantly higher than MINDS.

V. PERFORMANCE VALIDATION

The effectiveness of MINDS is validated through simulation. This section discusses the simulation setup, performance metrics and results.

A. Performance Metrics and Experiment Setup

The experiments are based on a 1500m × 1500m square area where random topologies are generated for varying number of segments '*n*' from 3 to 12, and communication range R for the collection points and MDCs is fixed at 100m. We consider the following metrics:

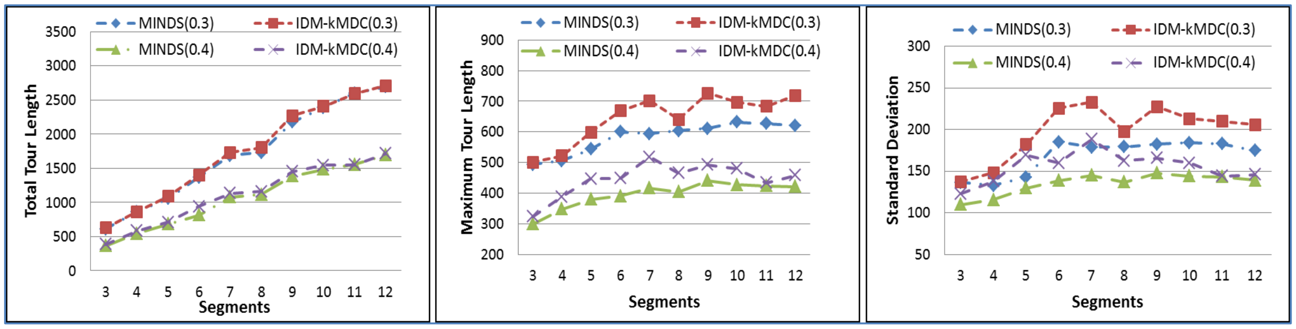
- *Total Tour Length*: is the sum of tour lengths of all employed MDCs. The goal is to minimize this metric in order to limit the overhead imposed on the MDCs.
- *Maximum Tour length*: is the largest among all MDCs tours. A large value indicates increased data delivery latency.
- *Standard Deviation*: indicates the dispersion from the average tour length. A low value indicates that the tour lengths are very close to the average and the MDCs are sharing the travel load equitably. A large value indicates unbalanced tours, which can cause delays in data distribution and result in bandwidth and storage issues for large data volumes or in case of high latency.

The performance of MINDS is compared to IDM-kMDC, which uses a bottom-up approach by forming *n*-1 tours, where *k* < *n*-1, and then merges some of these tours until the number of employed MDCs equals *k*. We study the performance while varying the following parameters:

- *Number of Segments 'n'*: This enables the evaluation of the scalability of the approach.
- *Number of Available MDCs* ($k = T_{Relays} * \phi$): This is the number of MDCs available in the disjoint network for connectivity restoration. The value of k is not changed directly since depending on the randomly generated configuration the available MDCs may be sufficient to form a stable inter-segment topology by acting as stationary RNs. To overcome this issue and enable meaningful comparison across a wide variety of configurations, we determine k based on the network layout. We first calculate the number of stationary RNs that are required to restore connectivity along the *mst* edges. The number of RNs required for an edge is given by $N_{Relays} = \lceil \frac{\text{Edge Length}}{R} \rceil - 1$ and the total number of relays T_{Relays} is the sum across all *mst* edges. To validate the performance of MINDS under different values of 'k', we set 'k' to be a fraction ϕ of T_{Relays} since having MDC count that exceeds T_{Relays} makes forming a stable inter-segment more appropriate than the use of tours. We are also primarily concerned with topologies where the value of k is less than the number of *mst* edges but greater than one, i.e., less than *n-1*, to highlight the difference between the tours constructed by IDM-kMDC and MINDS, since a higher k value or 'k=1' leads to the same result for both algorithms. In the experiments we capture the performance for ϕ = 0.3 and 0.4 since the value of k is found to meet the above constraints under these settings for most the generated topologies.

B. Simulation Results

This section discusses the obtained results. Each configuration is averaged over 100 different random topologies. We ignore results from topologies where *k*>*n*-1 as both algorithms have the same results. Figure 4 reports the effect of varying ϕ on the performance metrics. The results seen in Figure 4(a) indicate that MINDS and IDM-kMDC are closely matched in total tour length with MINDS performing slightly better in most cases. The biggest advantage of MINDS is showcased in Figure 4(b) where it is clear that the length of the maximum tour is noticeably smaller for MINDS than IDM-kMDC. So compared to IDM-kMDC for a topology having approximately the same total tour length, MINDS results in a smaller maximum tour length, this indicates that the load is better spread amongst the MDCs. This observation is further confirmed when comparing the standard deviation of the tours in Figures 4(c) MINDS clearly outperforms IDM-kMDC by having tours whose lengths are



(a) (b) (c)

Figure 4: MINDS vs IDM-kMDC for ϕ=0.3 and ϕ=0.4 (a) Total Tour Length, (b) Maximum Tour length; and (c) Std. deviation of Tours as a function of the number of segments.

closer to the average. This reiterates that the tours determined by MINDS are clearly better balanced while also minimizing the total tour length by matching the result of the greedy IDM-kMDC approach. The performance advantage of MINDS is sustained with increased availability of MDCs (larger value of ϕ).

Overall, the simulation results shown in Figure 4 confirm the performance advantage of MINDS as it imposes significantly less overhead on an individual MDC by having better balanced tours than IDM-kMDC while also minimizing the total tour length. This balancing prevents an MDC from dying out faster due to covering a larger distance than its peers, it reduces data latency and is less likely to suffer from memory or bandwidth issues compared to IDM-kMDC. MINDS is easily scalable and its performance advantage grows for large networks, as it has a lower time complexity than IDM-kMDC and thus converges faster.

VI. CONCLUSION

Internetworking of disjoint segments would be essential for sustaining the operation of a WSN after a major damage that affects multiple nodes. In this paper we have presented MINDS, a novel algorithm that enables restoring connectivity after failure by constructing balanced and optimized data collection tours using the available mobile nodes within the network. MINDS achieves this by initially forming a minimum spanning tree over the segments and successively partitioning the *mst* around the center until all available MDCs have been employed. The simulation results have demonstrated that the proposed algorithm outperforms competing approaches. Future extensions include dealing with connectivity restoration problems in which we have a mixture of stationary mobile relay nodes.

ACKNOWLEDGEMENTS: This work was supported by the National Science Foundation (NSF) award # CNS 1018171.

REFERENCES

[1] X. Han, X. Cao, E. L. Lloyd and C.-C. Shen, “Fault-tolerant Relay Nodes Placement in Heterogeneous Wireless Sensor Networks,” *Proc. of INFOCOM’07*, Anchorage AK, May 2007.

[2] N. Li and J. C. Hou, “Flss: a fault-tolerant topology control algorithm for wireless networks,” *Proc. of MobiCom ’04*, Philadelphia, PA, Sept 2004.

[3] X Wang, G Xing, Y Zhang, C Lu, R Pless and C Gill, “Integrated coverage and connectivity configuration in wireless sensor networks”, *Proc. of ACM Sensys’03*, Los Angeles, CA, November 2003.

[4] B. Chen, et al., “Span an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks,” *Proc. of ACM MobiCom’01*, Rome, Italy, July 2001.

[5] G. Wang, G. Cao, T. La Porta, and W. Zhang, “Sensor Relocation in Mobile Sensor Networks,” *Proc. INFOCOM’05,* Miami, FL, Mar. 2005.

[6] D. Cerpa, Estrin, “ASCENT: adaptive self-configuring sensor networks topologies,” *Proc. of the INFOCOM’02*, New York, NY, June 2002

[7] M. Younis, S. Lee, A. A. Abbasi, “A Localized Algorithm for Restoring Internode Connectivity in Networks of Moveable Sensors,” *IEEE Trans. on Computers*, 59(12), pp. 1669-1682, Aug. 2010.

[8] K. Akkaya, F. Senel, A. Thimmapuram, S. Uludag, “Distributed Recovery from Network Partitioning in Movable Sensor/Actor Networks via Controlled Mobility,” *IEEE Trans. on Comp.*, 59(2), pp.258-271, 2010.

[9] F. Senel, K. Akkaya and M. Younis, “An Efficient Mechanism for Establishing Connectivity in Wireless Sensor and Actor Networks,” *Proc. of Globecom’07*, Washington, DC, Nov. 2007.

[10]Y. K. Joshi and M. Younis, “Autonomous Recovery from Multi-node Failure in Wireless Sensor Network” *Proc. of Globecom’12*, Anaheim, CA, Dec. 2012.

[11]G. Lin, G. Xue, “Steiner Tree Problem with Minimum Number of Steiner Points and Bounded Edge-length,” *Information Processing Letters*, 69(2), pp. 53-57, 1999.

[12] X. Cheng, D.-z. Du and L. Wang and B. Xu, “Relay Sensor Placement in Wireless Sensor Networks,” *Wireless Networks*, 14(3), pp. 347-355, 2008.

[13] E. L. Lloyd, G. Xue, “Relay Node Placement in Wireless Sensor Networks,” *IEEE Trans. on Comp.*, 56(1), pp. 134-138, Jan 2007.

[14] F. Senel and M. Younis,” Optimized Relay Node Placement for Establishing Connectivity in Sensor Networks” *Proc. of Globecom’12*, Anaheim, CA, Dec. 2012.

[15] Yatish K. Joshi and M. Younis, "Distributed Approach for Reconnecting Disjoint Segments ", *Proc. of Globecom’13*, Atlanta, GA, December 2013.

[16] S. Lee, and M. Younis, “Recovery from Multiple Simultaneous Failures in Wireless Sensor Networks using Minimum Steiner Tree,” *Journal of Parallel and Distributed Computing*, Vol. 70, pp. 525-536, April 2010.

[17]F. Senel, M. Younis, “Optimized Interconnection of Disjoint Wireless Sensor Network Segments Using K Mobile Data Collectors,” *Proc. of Int’l Conf. on Comm. (ICC’12)*, Ottawa, Canada, Jun 2012.

[18]J. L.V.M. Stanislaus, and M. Younis, “Mobile Relays Based Federation of Multiple Wireless Sensor Network Segments with Reduced-latency,” *Proc. of Int’l Conf. on Comm. (ICC’13)*, Budapest, Hungary, June 2013.