

RESTORING CONNECTIVITY IN PARTITIONED WIRELESS SENSOR
NETWORKS

by

Izzet Fatih Senturk

B.S., Ege University, 2006
M.Eng., Cornell University, 2008

A Dissertation
Submitted in Partial Fulfillment of the Requirements for the
Doctor of Philosophy Degree

Department of Computer Science
in the Graduate School
Southern Illinois University Carbondale
December, 2013

DISSERTATION APPROVAL

RESTORING CONNECTIVITY IN PARTITIONED WIRELESS SENSOR NETWORKS

By

Izzet Fatih Senturk

A Dissertation Submitted in Partial

Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in the field of Computer Science

Approved by:

Kemal Akkaya, Chair

Bidyut Gupta

Qiang Cheng

Mengxia Zhu

Mohamed Younis

Graduate School
Southern Illinois University Carbondale
October 29, 2013

AN ABSTRACT OF THE DISSERTATION OF

IZZET FATIH SENTURK, for the Doctor of Philosophy degree in COMPUTER SCIENCE, presented on 29 OCTOBER 2013, at Southern Illinois University Carbondale.

TITLE: RESTORING CONNECTIVITY IN PARTITIONED WIRELESS SENSOR NETWORKS

MAJOR PROFESSOR: Dr. K. Akkaya

The sensor nodes in Wireless Sensor Networks (WSNs) and Mobile Sensor Networks (MSNs) can be prone to failures due to limited resources and/or the harsh environments where they are deployed. The network may be subject to partitioning if such failures are experienced by the cut-vertex nodes in the system. In case of partitioning, connectivity of the nodes in disjoint partitions with the sink node is disrupted. This not only affects the data delivery but also the possible cooperation and coordination of the nodes in handling certain events. To restore the connectivity of a partition with the rest of the network, network topology should be adjusted through either exploiting existing mobile nodes in the network or introducing additional relay nodes (RNs) to the network. However, both solutions pose certain challenges. In the former case, the mobility of the nodes requires significant energy consumption and thus the movement distance should be minimized. In addition, if the scope of the damage is too wide, determining the nodes to be relocated and their final locations is another challenge. In the latter case, determining the number of RNs and a self-configuring scheme for their movement destinations need to be tackled. In case of unavailability of sufficient RNs to provide connectivity with stable links to the whole network, another solution can be providing intermittent connectivity to the partitions by employing RNs as Mobile Data Collectors (MDCs). A mixed solution where some of the RNs are employed as MDCs and some as stationary RNs raises the challenge of determining

the number of stationary RNs and identifying their locations, assigning MDCs to serve partitions uniformly in such a way that the tour lengths of MDCs are minimized and the load among the MDCs are balanced.

In this dissertation, we address the connectivity restoration problem in partitioned WNSs and MSNs due to large scale damages. We present centralized and distributed approaches while considering four cases: 1. Minimizing the movement cost of the nodes while utilizing existing nodes in the network in case of the availability of the mobile nodes/actors. 2. Minimizing the number of relay nodes to be used and their movement cost in case of the lack of mobile nodes/actors in the network. 3. Maximizing the number of nodes served with a stable link while not exceeding the maximum tour length defined on MDCs when a mixed solution is required where some or all of the RNs are employed as MDCs. 4. Considering QoS constraints and rendezvous waiting time when multiple MDCs are in use. The effectiveness of all proposed approaches are validated through extensive simulation experiments.

DEDICATION

This work is dedicated to my mother, Hulya, my father, Yunus, and my sister, Esra, for their unwavering support and encouragement over the years.

ACKNOWLEDGMENTS

Words cannot express my gratitude to the individuals who paved the way for me to finish this journey. I am extremely grateful for Dr. Kemal Akkaya, my dissertation advisor at Southern Illinois University Carbondale, whose support, inspiration, guidance, and encouragement blessed my efforts and made it possible to finish my dissertation in three and one half years. My sincere thanks also goes to the other four members of my graduate committee, Dr. Bidyut Gupta, Dr. Qiang Cheng, Dr. Mengxia Zhu, and Dr. Mohamed Younis, for their patience, understanding, and sound wisdom during this process.

My parents have been encouraging and assuring in all aspects of completing my PhD and I would not have made it if it weren't for them who helped me along the way. So many others encouraged and prayed for me. I would like to take this opportunity to thank them to be with me with their prayers.

Finally, I would like to acknowledge that this work has been supported by US National Science Foundation under the grant number CNS 1018404. Any opinions, findings, and conclusions or recommendations expressed in this dissertation are those of the author and do not necessarily reflect the views of the National Science Foundation.

TABLE OF CONTENTS

Approval	i
Abstract	i
Dedication	iii
Acknowledgments	iv
List of Tables	xi
List of Figures	xii
List of Algorithms	xvi
1 Preliminaries and Research Problem	1
1.1 Introduction	1
1.2 Partitioning in the Network	1
1.3 Connectivity Restoration Using Existing Nodes	3
1.4 Connectivity Restoration Using Relay Nodes	4
1.5 Contribution	5
1.6 Organization	10
2 Related Work	11
2.1 Proactive Solutions	11
2.2 Reactive Solutions	12
2.2.1 Connectivity Restoration by Exploiting Existing Nodes	12
2.2.2 Connectivity Restoration by Introducing External Relay Nodes	14
2.2.3 Connectivity Restoration by Employing Relay Nodes as Mobile Data Collectors	16
3 An Optimization-based Approach for Connecting Partitioned Mobile Sensor/Actor Networks	17
3.1 Preliminaries	17
3.1.1 Assumptions	17

3.1.2	Problem Definition	18
3.2	Model	19
3.3	Performance Evaluation	22
3.3.1	Total Travel Distance	22
3.3.2	Delay	23
3.4	Summary	24
4	An Effective and Scalable Connectivity Restoration Heuristic for Mobile Sensor/Actor Networks	26
4.1	Preliminaries	26
4.1.1	Assumptions	26
4.1.2	Problem Definition	27
4.2	Approach	27
4.2.1	Overview	27
4.2.2	Phase I - Determining Target Locations	28
4.2.3	Phase II - Connecting Partitions	29
4.2.3.1	Connectivity Maintenance	29
4.2.3.2	Determining Specific Locations for Dominatees	30
4.2.4	Algorithm Pseudo Code and Analysis	32
4.2.4.1	Time Complexity Analysis	32
4.2.4.2	Total Movement Complexity Analysis	34
4.3	Experimental Evaluation	35
4.3.1	Performance Comparison with the Near-optimal Solution (OBA)	36
4.3.1.1	Total Movement Distance	36
4.3.1.2	Recovery Delay	37
4.3.2	Performance Comparison for Assessing Scalability	39
4.3.2.1	Total Movement Distance	39
4.3.2.2	Recovery Delay	40

4.4	Summary	42
5	Distributed Connectivity Restoration in Mobile Sensor Networks using Game Theory	43
5.1	Preliminaries	43
5.1.1	Assumptions	43
5.1.2	Problem Definition	44
5.2	Centralized Heuristic Approaches	44
5.2.1	Density Based Centralized Heuristic (DeCH)	45
5.2.2	Distance Based Centralized Heuristic (DiCH)	47
5.3	Game Theory Based Distributed Heuristic (GTDH)	49
5.3.1	Game Theory Background	49
5.3.2	Model	50
5.3.2.1	Overview and Notation	50
5.3.2.2	Reaching Equilibrium	52
5.3.3	Distributed Heuristic based on Game Theory	53
5.3.3.1	Approach Overview	53
5.3.3.2	Computing the Equilibrium and Probability Distribution	54
5.3.3.3	Algorithm Pseudo Code	57
5.4	Experimental Evaluation	58
5.4.1	Simulation Setup	58
5.4.2	Metrics	59
5.4.3	Baselines	60
5.4.4	Performance Comparison Including the Sub-Optimal Solution	61
5.4.5	Comparison Excluding the Sub-Optimal Solution	61
5.4.5.1	Total Travel Distance	61
5.4.5.2	Maximum Travel Distance	64
5.4.5.3	Coverage	66

5.5	Summary	67
6	Relay Placement for Restoring Connectivity in Partitioned Wireless Sensor Networks under Limited Information	68
6.1	Preliminaries	68
6.1.1	Assumptions	68
6.1.2	Problem Definition	69
6.2	Force Based Approach	70
6.2.1	Approach Description	70
6.2.2	Notation	72
6.2.3	Algorithm Analysis	72
6.2.3.1	Message Complexity	73
6.2.3.2	Computational Complexity	74
6.2.3.3	Movement Complexity	75
6.2.3.4	Convergence of the Algorithm	76
6.3	Game Theoretic Approach	77
6.3.1	Model	77
6.3.2	Algorithm and Pseudo Code	77
6.3.3	Algorithm Analysis	81
6.3.3.1	Message Complexity	81
6.3.3.2	Computational Complexity	82
6.3.3.3	Movement Complexity	82
6.3.3.4	Convergence of the Network	83
6.4	Experimental Evaluation	84
6.4.1	Simulation Setup	84
6.4.2	Experiment Results	86
6.4.2.1	Total Travel Distance	86
6.4.2.2	Maximum Travel Distance	88

6.4.2.3	Total Relays Deployed	90
6.4.2.4	Deployment Location	92
6.5	Summary	95
7	Connectivity Restoration in Disjoint Wireless Sensor Networks using Limited Num- ber of Mobile Relays	100
7.1	Preliminaries	102
7.1.1	Assumptions	102
7.1.2	Problem Definition	102
7.2	Approach	103
7.2.1	Overview	103
7.2.2	Iterative Second Phase	104
7.2.2.1	Clustering	104
7.2.2.2	Tour Bound Checking and Re-clustering	105
7.2.3	Determining Tour Path	106
7.2.4	Time Complexity Analysis	106
7.3	Experimental Evaluation	108
7.3.1	Experiment Setup	108
7.3.2	Evaluation and Analysis	108
7.3.2.1	Number of Sensors	108
7.3.2.2	Maximum Tour Length	110
7.3.2.3	Comparison with IDM-kMDC	111
7.4	Summary	112
8	Mobile Data Collector Assignment and Scheduling for Minimizing Data Delay in Partitioned Wireless Sensor Networks	114
8.1	Preliminaries	116
8.1.1	Assumptions	116
8.1.2	Problem Definition	117

8.2	Rendezvous Scheduling of MDCs for Reduced Delay	118
8.2.1	Background on MDC Tours	118
8.2.2	End-to-end Delay with a Single Sink	118
8.2.3	Rendezvous Scheduling for Reduced Delay	120
8.3	Delay-aware Clustering with Multiple Sinks	122
8.3.1	Motivation	122
8.3.2	Approach Overview	123
8.3.2.1	Determining Sink Locations	123
8.3.2.2	Clustering	124
8.4	Experimental Evaluation	125
8.4.1	Performance Metric and Baselines	125
8.4.2	Experiment Results and Analysis	126
8.4.2.1	Experiment Results with Single Sink	126
8.4.2.2	Experiment Results with Multiple Sinks	129
8.5	Summary	131
9	Conclusion and Future Work	132
9.1	Comparison of the Proposed Approaches	132
9.2	Future Work	133
	References	134
	Vita	142

LIST OF TABLES

5.1	Notation for the Proposed Approach	51
5.2	Optimality of the Given Approaches	61
5.3	Network coverage (m^2) with varying number of nodes when different approaches are applied.	66
5.4	Network coverage (m^2) with varying number of partitions when different ap- proaches are applied.	67
6.1	Notation for FBA	73
9.1	Comparison of the Proposed Approaches	133

LIST OF FIGURES

3.1	A partitioned WSAN.	18
3.2	Replacing the smallest circle containing a vector x with a regular polygon. . .	21
3.3	Error with the increased number of sides of a polygon.	22
3.4	Total travel distance for the three approaches when the number of nodes is varied.	23
3.5	Total travel distance for the three approaches when the number of partitions is varied.	24
3.6	Total travel distance for the three approaches when the number of nodes is varied.	25
3.7	Total travel distance for the three approaches when the number of partitions is varied.	25
4.1	An illustration of how CIST works. (a) Initial topology (b) Resulting topology of CIST where the location of gray nodes (RNs) will be used as target locations in Phase II.	29
4.2	CDS consisting of dominator nodes in a partition.	30
4.3	Illustration of Phase II. (a) Initial topology. (b) Final topology. Note that $d_1 < d_2$ and $d_4 < d_3$. Interface nodes $node_b$ and $node_e$ are not allowed to move. Note the increase in the number of dominatees after CDS update.	31
4.4	Worst case total movement before two partitions get connected. The distance from $node_1$ to the closest target location is $\frac{nr}{2}$	35
4.5	Total travel distance with 3 partitions and varying number of nodes.	36
4.6	Total travel distance with 20 nodes and varying number of partitions.	37
4.7	Maximum travel distance with 3 partitions and varying number of nodes.	38
4.8	Maximum travel distance with 20 nodes and varying number of partitions.	38
4.9	Total travel distance with 2 partitions and varying number of nodes.	39

4.10	Total travel distance with 150 nodes and varying number of partitions.	40
4.11	Maximum travel distance with 2 partitions and varying number of nodes.	41
4.12	Maximum travel distance with 150 nodes and varying number of partitions.	41
5.1	Estimating the Equilibrium	57
5.2	Network after recovery	58
5.3	Total travel distance with varying number of nodes, # of partitions is 2.	63
5.4	Travel travel distance with varying number of partitions, # of nodes is 150.	64
5.5	Maximum travel distance with varying number of nodes	65
5.6	Maximum travel distance with varying number of partitions	66
6.1	A partitioned WSN with a set of relay nodes.	70
6.2	The pushing forces applied to $Node_n$ by its neighbors (a), calculation of forces on $Node_n$ (b) and calculation of net force on $Node_n$ (c)	72
6.3	Scenario causing the worst case movement for FBA and GTA.	76
6.4	GTA may fail to ensure complete recovery for concave deployment.	84
6.5	Total travel distance with varying number of nodes. The number of partitions is fixed to 2.	88
6.6	Total travel distance with varying number of nodes. The number of partitions is fixed to 2. Results in Figure 6.5 is magnified for GTA and CA.	89
6.7	Total travel distance with varying number of partitions. The number of nodes is fixed to 150.	90
6.8	Total travel distance with varying number of partitions. The number of nodes is fixed to 150. Results in Figure 6.7 is magnified for GTA and CA.	91
6.9	Maximum travel distance with varying number of nodes. The number of partitions is fixed to 2.	92
6.10	Maximum travel distance with varying number of nodes. The number of partitions is fixed to 2. Results in Figure 6.9 is magnified for GTA and CA.	93

6.11	Maximum travel distance with varying number of partitions. The number of nodes is fixed to 150.	94
6.12	Maximum travel distance with varying number of partitions. The number of nodes is fixed to 150. Results in Figure 6.11 is magnified for GTA and CA. . .	95
6.13	The number of relays deployed with varying number of nodes. The number of partitions is fixed to 2.	96
6.14	The number of relays deployed with varying number of partitions. The number of nodes is fixed to 150.	97
6.15	Total travel distance with varying number of nodes. The number of partitions is fixed to 2.	98
6.16	Total travel distance with varying number of partitions. The number of nodes is fixed to 150.	98
6.17	The number of relays deployed with varying number of nodes. The number of partitions is fixed to 2.	99
6.18	The number of relays deployed with varying number of partitions. The number of nodes is fixed to 150.	99
7.1	A sample illustration for a partitioned WSN (a) and the planned RN locations that will reconnect the WSN as a result of CIST [43] algorithm (b)	103
7.2	The creation of clusters based on the number of RNs. Note that a partition may have multiple interface sensors and our approach considers them as separate locations as opposed to previous work. In the case of <i>partition</i> ₄ , we assume that one of the interface sensors can relay all the data it collects via routing to the other interface sensor so that the data can be forwarded to the sink.	105
7.3	(a) A disjoint WSN with MST points to be filled in. (b) A solution with 3 RNs and 1 MDC. (c) Another solution with 3 RNs and 1 MDC. The tour length in this case is less than that of (b) even though the same number of RNs are used in total.	107

7.4	Tour computation for 4 partitions with an interface sensor. The MDC will come close to the transmission range of each sensor to collect data.	108
7.5	# of connected sensors for varying partition count and L	109
7.6	# of connected sensors for varying partition count and m	110
7.7	Maximum tour length for varying partition count and L	110
7.8	Maximum tour length for varying partition count and m	111
7.9	Total tour length for different topologies.	112
7.10	Maximum tour length for different topologies.	112
8.1	A sample illustration of the occurrence of the waiting time in the collaborating MDCs	120
8.2	Clustering of partitions using Voronoi cells	124
8.3	Forming tours in the clusters	125
8.4	End-to-end delay of sensors for varying partition count. Node count: 100, MDC count: 2.	127
8.5	End-to-end delay of sensors for varying node count. Partition count:6, MDC count: 2.	128
8.6	End-to-end delay of sensors for varying MDC count. Node count: 50, Partition count: 6.	128
8.7	End-to-end delay of sensors for varying partition count. Node count: 100, MDC count for RAS : 2.	129
8.8	End-to-end delay of sensors for varying node count. Partition count: 6, MDC count for RAS : 2.	130
8.9	End-to-end delay of sensors for varying MDC count. Node count: 50, partition count: 6.	130

LIST OF ALGORITHMS

1	connect()	33
2	canMove($node_j$)	34
3	density_based_heuristic($partitions : Par_1, Par_2, \dots, Par_k$)	46
4	distance_based_heuristic($partitions : Par_1, Par_2, \dots, Par_k$)	48
5	computeEquilibrium($node_a$)	59
6	federation(Par_i)	60
7	FBA()	74
8	GTA()	79
9	schedule(MDC m)	122

CHAPTER 1

PRELIMINARIES AND RESEARCH PROBLEM

1.1 INTRODUCTION

Availability of low cost, tiny wireless sensors has enabled the deployment of wireless sensor networks (WSNs) comprising sensors in large quantities [1]. To take the advantage of mobility and sensing at the same time, mobile sensors, robots [2][3] and actuators [4] have been deployed in WSNs. Consequently, new types of WSNs have emerged such as Wireless Sensor and Actor Networks (WSANs) [4] which deploy mobile actors and Mobile Sensor Networks (MSNs) [5] which deploy mobile sensors. While MSNs can be used in applications with harsh environments such as environmental and habitat monitoring, WSAN applications include urban search and rescue, cyber-physical systems, forest fire fighting, destruction of mines in different environments. Due to their limited transmission range, the nodes are expected to collaborate in order to deliver their readings to the sink node via multi-hop paths. Unless deployed to pre-determined locations, random deployment of the nodes may leave node(s) isolated from the rest of the network due to initial deployment. Besides, limited energy supplies on sensors and their deployments in inhospitable environments (e.g., battlefield surveillance, forest fire detection, volcano or glacier monitoring and landslide detection, etc.) expose the nodes to failures which can cause severe partitioning in the network.

1.2 PARTITIONING IN THE NETWORK

Connectivity of the network is crucial to meet the desired application-level requirements such as coverage or delay tolerance in both MSNs and WSANs. In MSNs, all the sensors should remain connected with the rest of the network so that the sink node can keep receiving the data from the whole network. Similarly, in WSANs, actors collaborate with each other which requires to maintain an inter-actor connectivity at all times. How-

ever, the nodes in MSNs and WSANs are susceptible to failures due to several reasons such as limited on-board battery, electronic malfunctioning, or external damages due to the environmental conditions for being employed in hostile environments. The nodes may fail one at a time or simultaneously. Unless alternative paths are available, one or a set of nodes may be isolated from the rest of the network creating disjoint partitions in the network. Even a single node failure may create partitions, if the failed node was serving as a cut-vertex node in the network. Consequently, in MSNs some sensors will not be able to communicate with the sink and in WSANs, some actors will not be able to exchange data.

A proactive approach addressing the partitioning problem is decreasing the number of potential cut-vertex nodes by overloading the network through excessive redundant node deployment [6][7][8][9]. However, besides the additional node deployment cost and increased interference which may affect energy efficiency adversely, this approach cannot guarantee connectivity in large scale node failures. Thus, reactive approaches should be followed. Two types of schemes can be identified to restore connectivity in a partitioned network. The first approach is exploiting existing nodes if mobile nodes are available in the network. Using existing nodes avoids the deployment delay of additional nodes and provides a faster solution. On the other hand, since the location and the scope of the damage cannot be known in advance, all the nodes should be mobilized to guarantee a solution which may not be very cost efficient depending on the application and the failure rates. If the deployment delay is tolerable then another reactive approach can be utilized which proposes to introduce external relay nodes to the system. A relay node is a special node with more resources than a typical sensor. Relay nodes can be deployed to serve as a bridge in order to connect the partitions while the nodes in the partitions remain stationary. The approaches exploiting relay deployment can also be classified further: In the first group, the number of relay nodes to be deployed is assumed to be sufficient to assure network connectivity so that the relays remain stationary once the connectivity is ensured. On the other hand, the approaches in the second group assume the number of relay nodes may not be sufficient to guarantee

the connectivity of the whole network. Therefore, relays are employed as mules to carry data among the partitions so that connectivity can be provided through intermittent links. However, deployment of the nodes may not be possible at all in some application areas during certain times. Therefore, the approach to be applied should be picked carefully depending on the application requirements.

1.3 CONNECTIVITY RESTORATION USING EXISTING NODES

In this dissertation, the approaches proposed to address the problem of network partitioning is classified into two groups. The approaches studied in the first group assume mobile nodes are available in the system and can be exploited in order to restructure the network topology such that the connectivity of the nodes can be restored. The idea of utilizing existing mobile nodes instead of additional node deployment has attracted widespread attention given that this would be a self-healing process. In addition to WSNs, similar problem has been studied in WSANs and MSNs. Considering the fact that the movement requires significant energy comparing with the messaging cost, the main goal in these approaches is minimizing the movement cost. Since the occurrence of simultaneous partitions is possible, distributed schemes which can handle concurrent recovery process is preferable. A challenging issue in these efforts is determining the nodes to be relocated and the final location of the nodes after movement. Using the neighborhood information collected by the immediate (1-hop or 2-hop neighbors) neighbors, single node failures can be tolerated. However, determining the exact location for movement can be very challenging if multiple simultaneous node failures are considered assuming an alive node may be multiple hops away.

Even though, previous studies are available which claims to minimize the movement cost, they are limited to single node failures or cannot be applied to provide concurrent recovery. In this dissertation, we propose centralized and distributed novel approaches which can handle simultaneous node failures due to large scale damages. We also propose

a near optimal solution in Chapter 3 which can be used as a benchmark for centralized heuristics.

1.4 CONNECTIVITY RESTORATION USING RELAY NODES

The second group of approaches studied in this dissertation assumes external relay node deployment is possible in order to restore connectivity in a partitioned network. In this approach, the requirement of mobility for the existing nodes is waived and relay nodes is employed acting as connectors between different partitions of the network [10][11][12][13][14][15]. Typical relay node deployment heuristics studied before require a centralized approach and assumes either the location of partitions or the location of a partition representatives can be obtained. However, this is a strong assumption and the location of all the nodes may not be collected even though a centralized approach is used. Therefore, distributed approaches are required to guarantee connectivity with realistic assumptions. As opposed to previous studies which focus on centralized/manual placement of relays, our approach provides a distributed placement scheme once the relays are placed at a certain spot of the damaged region in a randomly manner. The goal is restoring connectivity by unifying the partitions while minimizing the number of relay nodes to be deployed and the movement cost of the deployed relay nodes.

Considering the fact that the scope of the damage may not be determined before the deployment of the relay nodes, the number of relays may not be sufficient to restore the whole network connectivity. In that case, instead of determining a permanent position for each relay, relays can be employed as mobile data collectors (MDCs). Each MDC is assigned a path designated to visit partitions, collect data from the partition, and pass the data to another partition. The objective in these approaches is determining the optimal paths for each MDC. Depending on the application requirements, minimizing travel cost and minimizing delay can be identified as the primary concerns while determining the optimal paths. The most common approach used in the previous studies is determining the

optimal spots for deployment assuming the number of relays is sufficient and then reducing the number of spots to the number of available relays while minimizing the tour lengths.

We present a novel distributed relay placement approach in Chapter 6 assuming the availability of sufficient relay nodes through multiple batch of deployments. Also we provide a novel relay placement approach in Chapter 7 assuming the unavailability of sufficient relay nodes to ensure the overall network connectivity. For the given relay nodes, some of the relays are employed as stationary nodes while the rest is employed as MDCs. The number of stationary nodes and their positions as well as the optimal paths for the MCDs to guarantee network connectivity is determined by the proposed approach.

1.5 CONTRIBUTION

In this dissertation, we studied the problem of connecting multiple disjoint partitions in a WSN through additional relay node deployment and using existing nodes. We propose 6 novel approaches which can guarantee connectivity in a partitioned network. The approaches can be summarized as follows:

- (a) *A Near Optimal Solution for Connectivity Restoration Using Existing Nodes:* The problem of federating multiple disjoint partitions through relocation of existing nodes in the network requires identifying the nodes to be relocated and their final locations. The node selection can be limited to the number of nodes in the network in the worst case. However, there are infinite number of possible positions to be visited in the application area when the location space is considered in real numbers. Thus, determining the optimal solution is a challenging problem and has been regarded as NP-hard and a near optimal solution is provided in [16] for WSANs. The proposed approach is modeled as a mixed integer programming problem with two different objective functions. The first objective function strives to minimize the total distance traveled by the nodes. On the other hand, the second objective function attempts to minimize the maximum distance traveled by any node participated to the recov-

ery effort. The second objective function implies the duration required to complete the recovery process. Besides, if an individual movement cost is too high, the node can be subject to an early battery depletion causing further partitions. The results obtained indicate the best performance known. However, the solution is not scalable and becomes infeasible when $n > 30$. This level of scalability can be acceptable for WSANs where the number of actors would be limited. On the other hand, considering a WNS with hundreds of nodes deployed, the approach will be inadequate to guarantee a solution in a timely manner. Nonetheless, this solution is the best solution in the literature in terms of movement distance and can be used as a benchmark for centralized heuristics.

- (b) *An Effective and Scalable Connectivity Restoration Using Existing Nodes:* For the problem of federating multiple disjoint partitions through relocation of existing nodes in the network, infinite number of possible positions to be visited in the application area should be considered in order to obtain an optimal solution. However, as mentioned earlier, considering such a large location space diminishes the scalability of the solution. In this dissertation, we propose an approach to reduce the number of possible locations where the nodes can visit while guaranteeing the recovery of the network connectivity. Initially, a set of locations is identified among infinitely many locations by running a relay node placement algorithm as proposed in [12]. The set of locations could be used to guarantee connectivity as if relay nodes were deployed to those locations. Nonetheless, assuming relay node deployment is not possible, existing nodes in the network are used to fill those locations. The nodes to be relocated are selected in such a way that the violation of the connectivity of the partition is avoided. To identify the nodes that will not cause further partitioning if removed, connected dominating set (CDS) of the partition is determined. Once the CDS of the partition is determined, the node is selected if it is not a member of the CDS

(i.e., dominatee). A greedy heuristic is followed to pick the dominatee node to be relocated if multiple dominatee nodes are available so that the movement distance can be minimized. Specifically, the distances between the set of unoccupied target locations and available dominatee node locations are sorted. The closest target location - dominatee location pair is selected and the dominatee node is relocated to the corresponding target location. Until the entire target locations are occupied, the process continues so that the connectivity is guaranteed. This solution performs very close to the near-optimal solution proposed in the first item and also scales well when the number of nodes or partitions are increased.

- (c) *A Distributed Game Theoretic Approach for Connectivity Restoration Using Existing Nodes:* The problem of minimizing the movement cost while re-establishing the connectivity in a partitioned network is addressed in a distributed manner assuming complete information about the post-failure network topology may not always be available in a realistic scenario. Proposed approach is based on Game Theory and unlike centralized approaches, the nodes to be moved and the direction of the movement is determined with limited knowledge on partitions and failure locations. Basically, nodes in each partition collaborate to assign an estimated Nash equilibrium for other partitions using a probabilistic model. Movement decision is made once the probability distributions are determined. Movement takes place towards the partition which has the dominant probability function until network wide connectivity is assured.

Unlike the near optimal solution [16], the presented approach is scalable and does not require complex computations while guaranteeing a solution and unlike previous centralized works it can be applied to scenarios where only partial information is available. Proposed approach facilitates former path directions before failure and node degrees which can be obtained locally. The proposed approach not only minimizes

the movement distance but also maximizes the network coverage while guaranteeing the recovery of the network connectivity.

- (d) *A Distributed Relay Placement Approach for Connectivity Restoration:* For the relay node placement problem, a distributed approach is presented in order to guarantee network recovery while minimizing the number of relay nodes to be deployed as well as the movement cost of the deployed relay nodes. Assuming once the relays are deployed randomly at a certain location in the damaged region, information regarding the damage scale and the remaining nodes will not be available. Therefore, relay nodes are expected to collaboratively decide the target locations for movement in order to restore the network connectivity with limited information. The proposed approach is based on Game Theory where movement decisions of the relays are regarded as a network game. Using limited information about the routes and partition boundary nodes, each partition is assigned an estimated Nash equilibrium and a probability distribution function (pdf). A partition is assigned a higher priority for recovery if it has a higher pdf. The recovery is pursued until all the partitions are connected and the same equilibrium is reached network-wide.

The proposed approach is scalable and suits well for problems where accurate information regarding the post-failure network state is not available. The number of relays to be deployed as well as the movement cost of the deployed relays can be minimized upon applying the presented approach.

- (e) *A Relay Placement Approach for Connectivity Restoration using Limited Number of Relays:* The connectivity problem of the network can be addressed through relay node placement if sufficient relay nodes are available. However, the scale of the failure may prevent assuring recovery for the whole network using the available relay nodes. Given this problem, we provide a solution by exploiting the motion capabilities of the relay nodes and employing them as MDCs to provide intermittent connectivity to

the partitions. Due to the excessive delay occurred by the mechanical movement, we strive to minimize the dependency on MDCs. On the other hand, employing fewer relay nodes as MDCs increases the movement cost of MDCs and delay. Therefore, a solution is required in order to determine the number of stationary RNs and MDCs so that the application requirements are met and the lifetime of the network is extended. Identifying the locations for stationary RNs and designating the tours for MDCs are other challenges to be addressed. Our solution guarantees connectivity by maximizing the number of stable connections while meeting the maximum tour constraint on the MDCs.

By initially employing all relays as MDCs, we follow an iterative procedure to reduce the MDC count while meeting the maximum tour length constraint. Specifically, groups of partitions are created and assigned to MDCs for touring. The proposed approach the simulation results indicate that the tour lengths are significantly smaller due to balanced clustering

- (f) *A Quality-of-Service Aware Solution for Mobile Data Collector Assignment and Scheduling:* The problem of restoring connectivity of a partitioned network can be addressed by employing MDCs to provide intermittent connectivity for partitions. In order to minimize the tour lengths, MDCs avoid to visit the sink directly and instead collaborate to pass data to the sink through other MDCs. Considering limited buffer of the nodes in the partitions and excessive amount of data to be transferred, MDCs should wait at rendezvous points to pass their data to the next MDC. This introduces rendezvous waiting besides the delays resulting from the touring. We present a novel scheduling algorithm to reduce the rendezvous waiting delay based on the tour lengths. We also present a solution assuming multiple sinks in order to eliminate the rendezvous waiting delay completely. The idea is clustering the partitions using p -center optimization and Voronoi cells based on the number of MDCs and assign-

ing each MDC to a cluster. Thus, the dependency of an MDC to another MDC to relay its data is eliminated and also the tour lengths of MDCs are balanced which minimizes the end-to-end delay significantly.

1.6 ORGANIZATION

The motivation and the background information for the problem of connectivity restoration in a partitioned network is provided in this chapter. The rest of the dissertation is organized as follows: Chapter 2 presents the related work regarding the problem motivated in Chapter 1. Chapter 3 discusses a near optimal solution which can also be used as a benchmark for centralized heuristics proposed for similar problems. Chapter 4 presents an efficient method to restrict the locations to be visited while relocating the nodes in a recovery scheme. Chapter 5 provides a semi-distributed approach for addressing the connectivity problem by using game theory. Chapter 6 discusses a relay placement approach assuming availability of sufficient relay nodes possibly through multiple batch deployments. Another relay placement approach is presented in Chapter 7 by employing some of the relays as MDCs to overcome the limitations on the number of available relays. Chapter 8 presents a QoS aware solution to avoid rendezvous waiting time for MDCs. Chapter 9 summarizes the proposed approaches and outlines future research plan.

CHAPTER 2

RELATED WORK

The published studies related to the contribution of this dissertation are discussed in this chapter. The related studies have been classified into two groups as proactive and reactive approaches. Reactive approaches are classified further into two groups. In the first group we discuss the approaches which strive to recover connectivity using existing nodes in the network. In the second group, approaches introducing external relay nodes to the system in order to restore network connectivity are discussed.

2.1 PROACTIVE SOLUTIONS

Considering the fact that network partitioning is caused by the removal of the node(s) serving as cut-vertex in the network, some approaches [6][7][8][9] propose to deploy excessive number of redundant nodes so that the number of cut-vertex nodes can be minimized. However, problems exist associated with this approach. First of all, redundant nodes require additional hardware and deployment cost which may not be neglected. Also, redundant nodes can cause interference which affects the success rate of the data delivery in case of retransmissions which incurs additional energy cost. Consequently, battery of the nodes can be depleted prematurely due to the adverse impact on the energy efficiency. In the worst case, this approach cannot guarantee a solution depending on the scale of the failure. If a large scale failure is experienced, node redundancy may not be high enough to compensate the failures. Due to the autonomous nature of the WSNs, reactive solutions should be pursued instead of precautionary approaches.

2.2 REACTIVE SOLUTIONS

2.2.1 Connectivity Restoration by Exploiting Existing Nodes

Using existing mobile nodes to recover connectivity in a partitioned network where the communication is disrupted has attracted widespread attention given that this would be a self-healing process. The main idea is exploiting mobility to restructure the network topology in a partitioned network by moving a set of nodes so that no nodes will be left isolated from the rest of the network. In WSNs, connectivity among the nodes is strived to be restored in order to assure network connectivity. On the other hand, in WSNs restoring inter-actor connectivity is the main goal. Depending on the network type (i.e., WSN, WSN, MSN), design (i.e., centralized, distributed), application requirements (i.e., delay tolerance) and the scale or the cause of partitioning, an approach may not be able to restore connectivity in a certain case while being able to handle in others. Partitioning can be caused by the failure of a single node or multiple nodes (i.e., simultaneous or not), or due to initial deployment. [17][18] offer recovery to partitioned networks which have experienced single node failures (i.e. one failure at a time). These approaches propose to replace a failed node by its neighbors. The nodes can obtain the location information of the neighbors locally and will be aware of the failure of a neighbor upon its silence after waiting a certain amount of time. The goal is replacing the failed neighbor with the least movement cost and without causing further partitions. The problem with these approaches is their limited scale compensation to failures. If a large scale failure occurs causing the failure of multiple nodes simultaneously, the next alive node may be multiple hops away and the location of the failed nodes to be replaced cannot be determined.

Block movement is pursued in [19] in order to restore connectivity in a partitioned network. The idea is moving the whole partition as a unit while conserving the formation towards another partition until two partitions connected. Though the idea is simple, minimum movement cost cannot be attained due to redundant node movements if this approach is applied. Unlike block movement, nodes are moved in a non-coordinated manner

in sequential movement. Upon the movement of a node, a further movement by a different node may be required due to the detachment of the neighboring nodes causing cascaded movement as in [20][21][22] so that the connectivity can be maintained. Among multiple candidates, [20] picks a node with fewer neighbors to limit the scope of the cascaded movement. [21] on the other hand, checks the connected dominating set of the partition and picks a dominee node if possible. [22] also considers delay and identifies the smallest disjoint blocks to reduce the recovery cost. The limitation of these approaches is their applicability to only the problems with single node failures assuming the location of the failed nodes would be available.

[23] and [24] can handle multiple node failures but [23] cannot provide a concurrent recovery scheme. Each failure is handled separately and a lock mechanism is used when a race condition occurs. Also, the locations of the failed nodes are assumed to be known by other nodes which is a strong assumption to be satisfied. On the other hand, [24] focuses on bi-connectivity and exploits attractive and repulsive forces to move the nodes towards the sink node. The location of the sink node is assumed to be known by other nodes. The advantage of this approach is its applicability to partitioned networks due to initial deployment since it does not require the availability of the initial locations of the nodes which can be difficult to obtain without the data transmission starts. On the other hand, this approach incurs higher movement cost comparing with the other approaches for the same scenarios. Especially, if the failure occurs very close to the sink node, excessive number of movements can be triggered.

[25] provides a solution assuming only two partitions are formed in a WSN. The sensors on the other hand are assumed to be connected network wide so that the failed node locations can be obtained by the moving actors. The goal of this approach is selecting the closest nodes from each partition and connect them by the mobile actors. The limitations and strong assumptions of this approach restrict its applicability to problem domains in a wider range. A centralized solution is presented in [26] which can handle multiple node

failures in a partitioned network. The approach considers connectivity restoration as an optimization problem and proposes a solution based on integer linear program (ILP). The idea is minimizing the movement cost while picking the nodes to be relocated and determining their final locations and providing a connected network at the end. This approach considers only the former locations of the failed nodes for the possible locations so that the optimal solution cannot be attained due to neglecting the infinite set of possible positions in the application area. Also, centralized behavior of the approach and considerably high computational complexity of the optimization problems are the limitations of this approach.

2.2.2 Connectivity Restoration by Introducing External Relay Nodes

If deployment delay is not an issue and the intervention to the application area is possible, another solution to the problem of connectivity restoration in a partitioned network is introducing external relay nodes to the system. A relay node is regarded as a special node with more resources (i.e., battery, transmission range) than a typical sensor. The advantage of this approach is eliminating the requirement of having mobile nodes in the network to restructure the network topology. If the size and the shape of the damaged region can be assessed, sufficient number of relay nodes can be deployed to the application area so that the relay nodes can link the partitions (i.e., acting as connectors between the partitions) as in [11][12][27][28][29][30] once a failure causing a permanent partitioning is detected. The goal in these approaches is minimizing the number of relay nodes to be deployed considering the labor and hardware cost. DORMS [11] presents a distributed algorithm for optimized relay node placement. The idea is moving the relay nodes from each partition towards the center of the application area assuming the locations of the partitions cannot be obtained in a distributed environment. An algorithm called FeSTA is given in [12]. In FeSTA, each partition is represented by a terminal. By forming a triangle by each subset of 3 terminals, an optimal solution is pursued connecting 3 partitions. In

[27], the lifetime of the network is investigated when mobile nodes are employed as mobile relays and mobile sinks. Two versions of relay node placement problems are studied in [28]. While the goal is minimizing the number of relay nodes for deployment, in the first version, connecting paths between the pair of sensors are assumed to be a mix of relay and sensor nodes. In the second version, connecting paths are assumed to be consisting only relay nodes. [29] assumes heterogeneous wireless sensor networks where the nodes may have different transmission ranges. Through the deployment of relay nodes, 2-connected network is provided in [30] where every sensor is able to communicate with at least two relay nodes.

In order to determine the minimum number of relay nodes to be used, typically previous studies proposed using Minimum Steiner Tree (MST). An MST is a tree which connects a number of points in two-dimensional space with the least number of nodes in the tree. Specifically, each partition is represented by a point and each relay node corresponds to a node of MST. Given that this problem is NP-Hard [11], most of these heuristics strive to improve the approximation ratio in terms of relay node count. The problem with the relay placement approaches rises when the intervention to the deployment area is not convenient and accessing the damages region is not possible. This causes the failure of assessing the scope of the damage and leaves the approaches, requiring detailed knowledge of the number/location of the failed nodes, impractical since they would fail to perform a pre-computation to determine the relay node locations and their count. Distributed approaches supporting self-deployment is required in order to guarantee a solution at all times without the knowledge of the damage scope. Relay nodes can be deployed at a certain safe location of the damaged area (i.e., via unmanned aerial vehicle) and then reposition themselves in a collaborative manner to guarantee connectivity. Another goal to be satisfied should be minimizing the movement cost of the relay nodes so that they can operate for a longer time.

2.2.3 Connectivity Restoration by Employing Relay Nodes as Mobile Data Collectors

Employing relay nodes as mobile data collectors (MDCs) is a special case of introducing external relay nodes to the network. If the number of relays is not sufficient to connect all the partitions in the network, relays can be employed as mules carrying data among the partitions [31][32][33][34][35]. MDCs connect the disjoint partitions with intermittent links in these approaches. Two main goals can be identified: minimizing the tour lengths of the relays and minimizing delay incurred due to travel time. Depending on the algorithm design, a combination of the mobile and stationary relays can also be used where the stationary relays permanently occupies an optimal location determined to satisfy the application requirements while the mobile relays constantly moves on the determined path.

Previous works either considered all the RNs as mobile and provided intermittent connectivity for all the nodes or considered a mixture of stationary and mobile RNs. In the first category of works, Senel and Younis consider using k MDCs where all the links will be intermittent. They propose a polynomial time heuristic for interconnecting Disjoint Segments with k MDCs (IDM-kMDC) with the goal of minimizing the maximum tour length of a MDC [36]. A similar work is done in [37]. In their solution, the employed mobile RNs keep moving in the network based on a certain policy to provide the intermittent connectivity over a time period instead of continuous connectivity.

The second category of works restricts the number of MDCs. For instance, [38] assumes mobile RNs where some of these can be placed as stationary. The goal is to restore connectivity with reduced inter-partition data delay. Another similar work is proposed in [39] by initially assuming a certain number of stationary and mobile RNs. The goal of the approach is to minimize the tour length of the mobile RNs by deploying the fixed RNs at certain locations. Once an SMT is computed, the found SPs and terminals are clustered such that a mobile RN will serve a single cluster. Therefore, the number of the formed clusters is restricted by the number of mobile RNs.

CHAPTER 3

AN OPTIMIZATION-BASED APPROACH FOR CONNECTING PARTITIONED MOBILE SENSOR/ACTOR NETWORKS

In this chapter, we present a mathematical model which minimizes the total travel distance while ensuring the federation of disjoint partitions in a mobile sensor/actor network. The approach is based on transportation network flows such that each node will receive one supply out of $n - 1$, distributed by one of the nodes in a network comprised of n nodes once the network is connected. The problem is modeled as a mixed integer nonlinear program and the nonlinear terms in the model are linearized using a polygon approximation. A second objective is also explored in order to minimize the maximum travel distance of any node. Maximum distance traveled by the nodes implies the delay required to complete the recovery operation and should also be minimized. We claim that the solution presented in this chapter can be used as a benchmark for other heuristics.

This chapter is organized as follows: Assumptions and the problem definition are given in Section 3.1. The model is presented in Section 3.2. The validation experiments and simulation results are provided in Section 3.3 and the chapter is concluded in Section 3.4.

3.1 PRELIMINARIES

3.1.1 Assumptions

We assume a WSAN where sensors and actors are deployed together. The actors are assumed to be less resource restricted (i.e., battery, transmission range) and have the ability to move on demand. The actors are also assumed to be able to locate themselves in the application area through mechanisms like GPS or other means [40]. Overlay network of the actors, which is called inter-actor network, can be partitioned due to initial deployment or massive failures as depicted in Fig. 3.1. Since the connectivity with the healthy actors in remote partitions will be lost, command center (i.e., sink) cannot obtain information about

such healthy actors and take necessary actions. To overcome this problem, an Unmanned Aerial Vehicle (UAV) is assumed to be available to collect information about the healthy actors. Once the post-failure network topology is obtained, recovery scheme can be applied. An obstacle free environment is assumed where a straight path is always available from a given source to a destination. Note that, actor and node is used interchangeably throughout the chapter when we refer to the individual elements of the network.

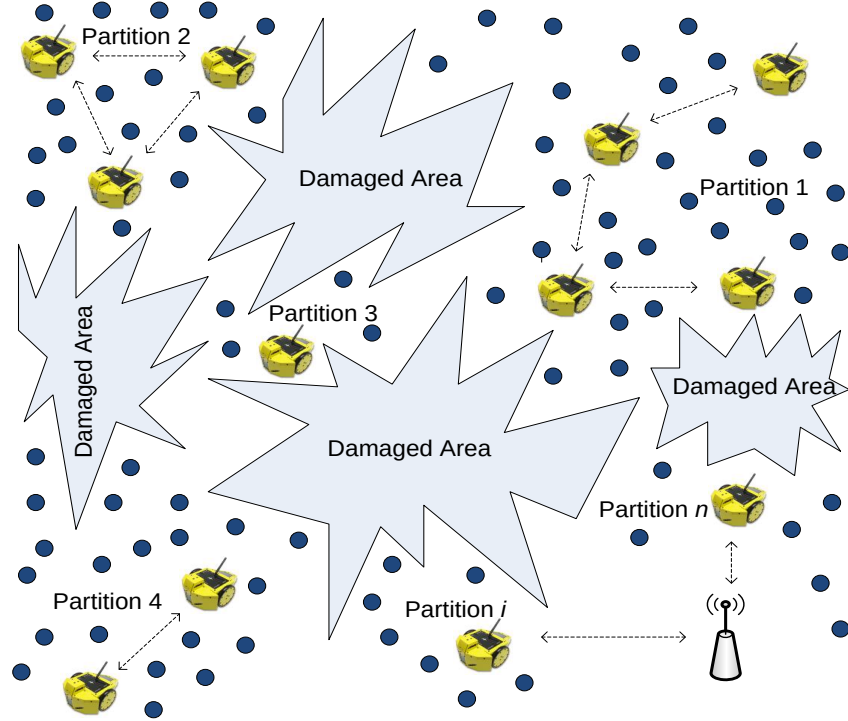


Figure 3.1: A partitioned WSN.

3.1.2 Problem Definition

Our problem can formally be defined as follows: Consider a WSN with n identical actors, a transmission range of r and multiple partitions, i.e., not all the actors in the network can communicate with each other. Let $\mathcal{A} = \{1, 2, \dots, n\}$ be the set of actors and

$\mathbf{p}^i = \begin{bmatrix} l_1^i \\ l_2^i \end{bmatrix}$ be the location coordinates of actor i , $i \in \mathcal{A}$. Our problem is to move the actors in the network so that the WSN becomes connected (i.e., an actor can communicate with

all other actors either directly or through other actors) while minimizing the total travel distance of all actors (i.e., $\sum_{i \in \mathcal{A}} d_i$) where d_i is the distance traveled by A_i .”

Connectivity of the WSA is defined as follows: Let $\mathbf{x}^i = \begin{bmatrix} x_1^i \\ x_2^i \end{bmatrix}$ be the position coordinates of actor i , $i = 1, 2, \dots, n$, after it is relocated. Connectivity between actor i and j are determined by the following variable:

$$y_{ij} = \begin{cases} 1, & \| \mathbf{x}^i - \mathbf{x}^j \| \leq r, \\ 0, & \text{otherwise.} \end{cases}$$

Given the infinitely many number of locations where the actors can be relocated, the problem is similar to Minimum Steiner Tree problem (MST) which connects k partitions using the least number of relay nodes. MST has been shown to be NP-Hard [41] and our problem is related to MST since once the relay nodes’ locations are determined when finding MST, the appropriate actors can be relocated to those locations afterwards and thus the network becomes connected again.

3.2 MODEL

We model the connectivity problem using a flow-based approach. In other words, we view the wireless communication network between actors as a transportation network, where a direct communication between two actors is viewed as a bidirectional transportation link between those actors. We assume that this “transportation” network can be used to transport items from one actor to another. One of the actors, chosen arbitrarily, assumed to have $n - 1$ items to supply (this actor will be referred to as the “source” actor) and that every other actor requires exactly one item. Let f_{ij} be the amount of flow on the transportation link from actor i to actor j . If the flow model is feasible (i.e. every actor receives an item as a result of the flows) then the entire network is connected. Otherwise,

the network must have partitions and therefore the items cannot be delivered to every actor in the network. We consider two objectives: 1) Minimizing total distance traveled by all actors; and 2) Minimizing maximum distance traveled by any actor.

Letting M be a very large positive number and i^* be the index of the source actor, the flow-based formulation for optimally relocating actors to restore connectivity can be written as:

$$\min \sum_{i \in \mathcal{A}} \|i - \mathbf{p}^i\| \quad (3.1)$$

$$\min \max_{i \in \mathcal{A}} \|i - \mathbf{p}^i\| \quad (3.2)$$

subject to

$$\|i - \mathbf{x}^j\| \leq r y_{ij} + M(1 - y_{ij}), \quad \forall i, j \in \mathcal{A}, \quad i \neq j, \quad (3.3)$$

$$\sum_{j \in \mathcal{A} \setminus \{i^*\}} f_{i^*j} = n - 1, \quad (3.4)$$

$$\sum_{i \in \mathcal{A}, i \neq j} f_{ij} = 1 + \sum_{k \in \mathcal{A}, k \neq j} f_{jk}, \quad \forall j \in \mathcal{A} \setminus \{i^*\}, \quad (3.5)$$

$$f_{ij} \leq (n - 1)y_{ij}, \quad \forall i, j \in \mathcal{A}, \quad i \neq j, \quad (3.6)$$

$$i \in^2, \quad i \in \mathcal{A}, \quad (3.7)$$

$$y_{ij} \in \{0, 1\}, \quad i, j \in \mathcal{A}, \quad i \neq j, \quad (3.8)$$

$$f_{ij} \geq 0, \quad i, j \in \mathcal{A}, \quad i \neq j. \quad (3.9)$$

As mentioned earlier objective functions (3.1) and (3.2) are to minimize total distance traveled by all actors and maximum distance traveled by any actor, respectively. Constraint (3.3) determines whether a pair of actors communicates directly with each other based on the distance between their locations. In this set of constraints, if y_{ij} is 1 then the distance between the actors i and j is forced to be less than r . On the other hand, if y_{ij} is 0 the constraints become non-binding since M is a very large number. Constraint (3.4) ensures

that $n - 1$ items are *shipped out* of the source actor, which is chosen arbitrarily, to be delivered other actors in the network. Constraints (3.5) are the flow-balance constraints which ensure that the total *flow into* an actor is equal to one item retained by that actor plus the total *flow out* of that actor. Constraints (3.6) ensure that the flow from one actor to another is zero if these two actors do not communicate directly. Finally, the actors are allowed to move anywhere on a two-dimensional plane (constraints (3.7)); y_{ij} variables representing direct communication between two actors are required to be binary (constraints (3.8)); and the flow variables f_{ij} are all positive (constraints (3.9)).

As final note, the formulation given above (Equations (3.1)-(3.9)) is a mixed integer nonlinear program (MINLP). MINLPs are known to be among the most difficult optimization problems. Therefore, for computational efficiency, the Euclidean distances used in objective functions (3.1) and (3.2) and constraints (3.3) are linearized using a polygon approximation described in [42]. The polygon approximation is based on the observation that the Euclidean norm of a vector (i.e., its length) is equal to the radius of the smallest circle centered at the origin containing the entire vector. This circle is replaced by a regular polygon centered at the origin (see Fig. 3.2). In this approximation scheme, the Euclidean norm of a vector is approximated by the apothem (i.e., the distance from the center to a side) of the smallest polygon containing the entire vector.

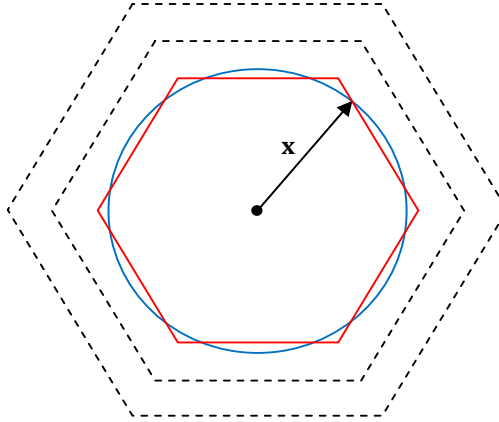


Figure 3.2: Replacing the smallest circle containing a vector x with a regular polygon.

The error that results from this approximation is small and decreases as the number of

polygon edges increases as shown in Fig 3.3. Numerical results show that the approximation error is negligible with polygons more than 20 edges.

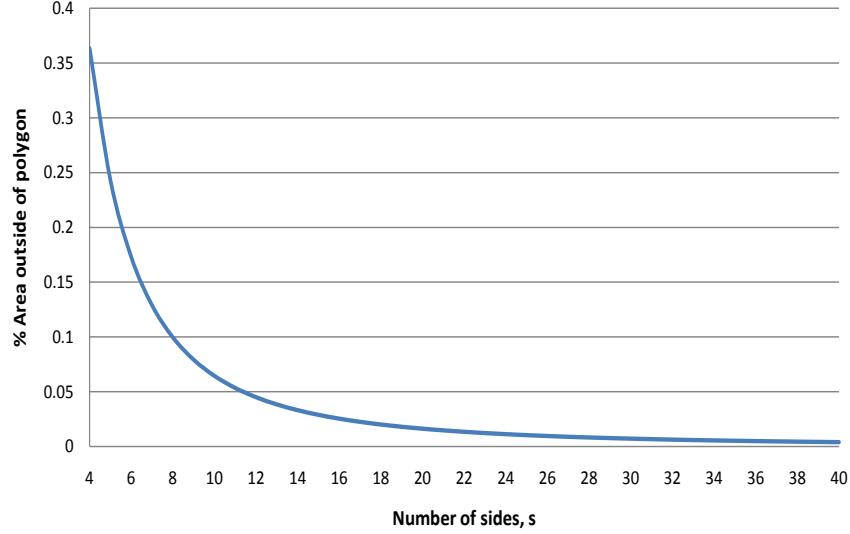


Figure 3.3: Error with the increased number of sides of a polygon.

3.3 PERFORMANCE EVALUATION

We evaluate the performance of the proposed approach regarding total travel distance and delay in this section by varying the number of nodes and partitions in the network.

3.3.1 Total Travel Distance

The results are depicted in Figures 3.4 and 3.5 for the total distance to be traveled when the proposed approach and the baseline approaches are applied. It can be observed that the least travel distance is incurred when the optimal solution is applied.

As expected, total travel distance increases when the number of partitions is increased in the network. This is due to the increased need for recovery of the nodes in partitions. On the other hand, total travel distance declines when the number of nodes is increased. In more populated networks, the partitions are expected to be denser where the distance between the partitions will be less in average thus, reducing the total travel distance of the recovery.

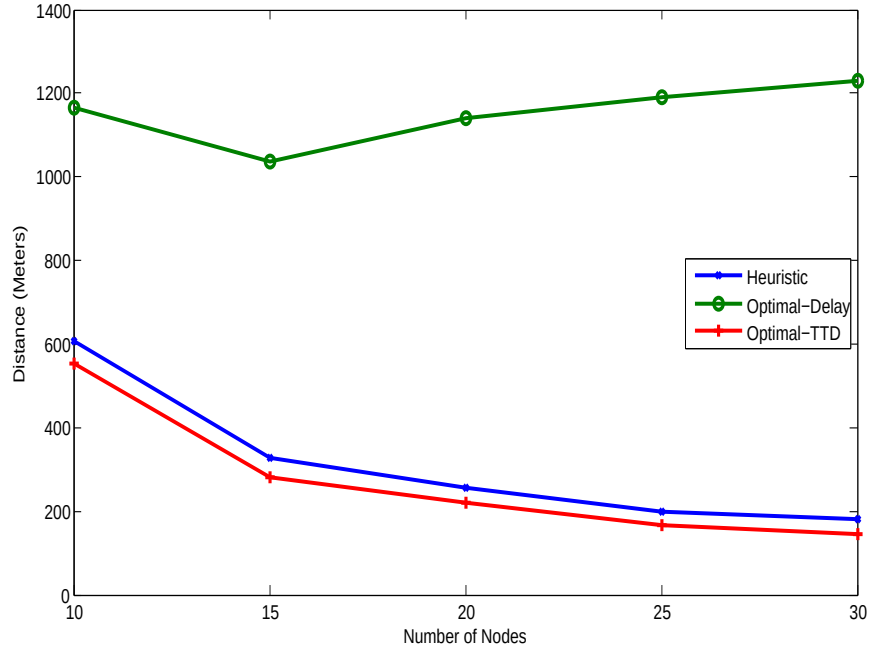


Figure 3.4: Total travel distance for the three approaches when the number of nodes is varied.

3.3.2 Delay

The results of the experiments for the connectivity restoration delay are shown in Figures 3.6 and 3.7. The optimal solution with the second objective provides the least delay. Our approach with the first objective also outperforms the heuristic while performing close to the optimal solution with the second objective.

With the increased number of nodes, the time for the recovery is decreasing given that the partitions are becoming much closer to each other. However, with the increased number of partitions there is no regular behavior of the approaches since the number of partitions will not affect the maximum travel distance of all the actors.

In summary, the optimal solution with the first objective provides the best total travel distance and performs better than the heuristic in terms of delay. If the delay is the sole concern, the second objective should be used. However, if both delay and energy are of concern, then the first objective will suit better. Obviously, depending on the application a weighted combination of both objectives can also be studied.

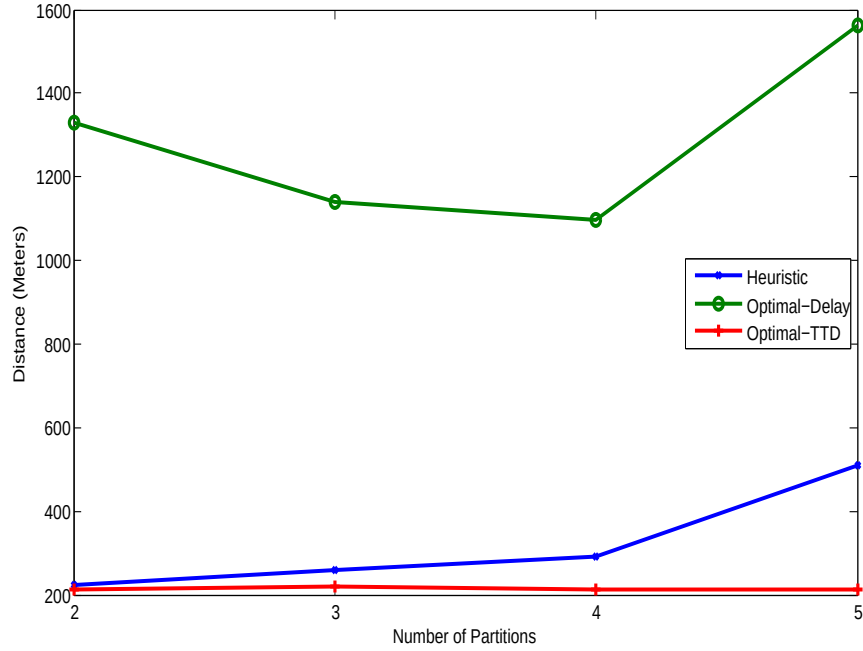


Figure 3.5: Total travel distance for the three approaches when the number of partitions is varied.

3.4 SUMMARY

In this chapter, we have presented a centralized near optimal solution to restore connectivity of a network (i.e. WSAN) with multiple disjoint partitions occurred due to massive node failures. The problem is modeled as a mixed integer program considering the connectivity of the nodes as transportation network flows to distribute a supply to each node in the network. Non-linear terms in the problem are linearized through a method called polygon approximation. The proposed approach is assessed through experiments revealing that the optimal solution in terms of travel distance can be achieved with a reasonable delay in terms of recovery time. We claim that the solution presented in this chapter can be used as a benchmark for other centralized heuristics.

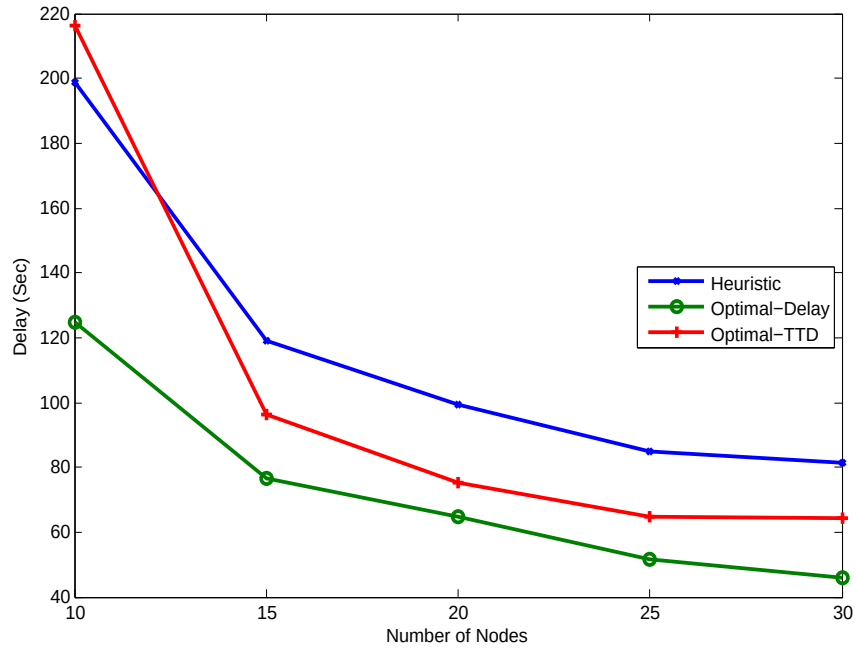


Figure 3.6: Total travel distance for the three approaches when the number of nodes is varied.

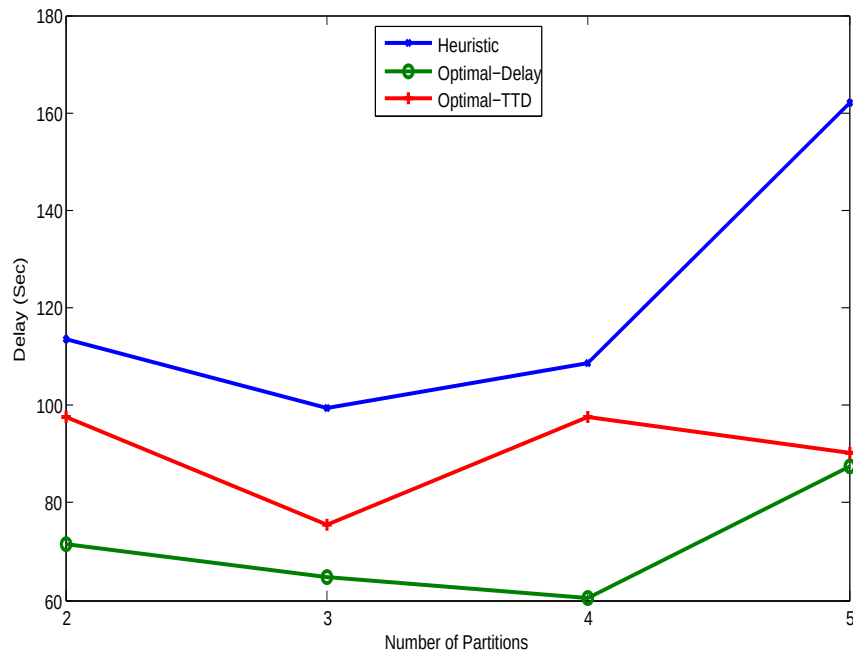


Figure 3.7: Total travel distance for the three approaches when the number of partitions is varied.

CHAPTER 4

AN EFFECTIVE AND SCALABLE CONNECTIVITY RESTORATION HEURISTIC FOR MOBILE SENSOR/ACTOR NETWORKS

In this chapter, an efficient yet scalable solution is proposed for the problem of connectivity restoration in a partitioned network using existing mobile nodes. As mentioned in Chapter 3, infinite number of positions should be considered while determining possible destination locations to obtain the optimal movement cost. However, handling such a large number of locations demands excessive computational power and offers limited scalability. The approach we present in this chapter addresses this problem and strives to minimize the number of possible locations to be visited while maximizing the movement cost efficiency as close as the optimal solution. The idea is identifying the optimal spots to place external relay nodes as if relay nodes will be introduced to the system. But, relay nodes are never deployed and these locations are used as potential destinations for the existing nodes. Extensive evaluations have revealed that the proposed approach performs very close to the near optimal solution in Chapter 3 while providing a better scalability.

This chapter is organized as follows: Assumptions and the problem definition are given in Section 4.1. The approach is presented in Section 4.2. The validation experiments and simulation results are provided in Section 4.3 and the chapter is concluded in Section 4.4.

4.1 PRELIMINARIES

4.1.1 Assumptions

We assume a partitioned network (i.e., WSAN, MSN) due to the initial deployment or a massive failure of the network elements (i.e., actors in WSANs, sensors in MSNs or nodes in general). We assume these network elements can move on demand without encountering any obstacles on the movement path. In addition, the nodes are assumed to know their locations through mechanisms like GPS. In order to identify healthy nodes in remote par-

titions, we assume a command center or an on-site temporary vehicle can communicate with the nodes via range extension or other means as a one-time event when the damage occurs. Thus, post-failure network topology can be obtained so that the recovery scheme can be applied.

4.1.2 Problem Definition

Formally, our problem can be defined as follows: “Given a set S of locations for n nodes where a node is represented by $node_i$ and has a transmission range of r . The nodes form k disconnected partitions and the number of nodes at each partition is at least 1. The goal is to provide a solution which will ensure that k partitions will be connected within a unique federated network (i.e., any node $node_i$ will be able to communicate with any of the $n - 1$ nodes in the network) by relocating the nodes such that the total travel distance of all the nodes (i.e., $\sum_{i=1}^n d_i$) is minimized where d_i is the distance traveled by $node_i$ ”.

4.2 APPROACH

4.2.1 Overview

In this chapter, we present a centralized heuristic to restore connectivity in a partitioned network by using the existing mobile nodes in the network. The idea is limiting the number of possible final locations for movement while minimizing the movement cost so that a scalable yet less computationally complex approach can be offered. The approach has two phases as detailed next. In the first phase, we determine the target locations by running a relay node placement algorithm which minimizes the relay node count. Relay node locations found as a result of running the relay node placement algorithm are used as the target locations. In the second phase, we exploit existing nodes to fill the relay node locations obtained in the first phase. When selecting the mobile nodes to fill these locations, we pick the closest ones to the damaged area whose movement does not create further partitioning within their respective partitions. The details of the phases are

elaborated next.

4.2.2 Phase I - Determining Target Locations

In the first phase, we use a relay node placement algorithm, named Connected Inter-Segment Topology (CIST) [43], to identify the set of target locations which can guarantee connectivity of the partitions by using the least number of relay nodes. CIST is a relay node (RN) placement heuristic which can restore connectivity of a partitioned network while minimizing the number of required RNs. Since the problem of RN placement in order to connect partitions is an NP-Hard problem [43], heuristics are pursued with a goal of minimizing the RN count while guaranteeing connectivity of the partitions. CIST is the best known heuristic in terms of its RN count and thus it is employed to reduce the number of target locations that will be correspond to RN locations.

CIST uses segment representation as a means of optimization. In CIST, unlike contemporary schemes, the network segments are not represented by a single node. Instead, all nodes which are located on the boundary of the individual segment are considered as segment representatives. Representing a segment with multiple nodes provides flexibility in finding more interface points (i.e. the sensor where RNs are attached) to make the segment connected. The algorithm has two steps. In the first step, CIST determines the minimal spanning tree (mst) edges between segments and the corresponding number of RNs required to establish connectivity on these edges. CIST then considers all combinations of three segments that include two segments on an mst edge. For each of these combinations, CIST determines the least RNs needed to form Steiner minimum tree with minimal Steiner points for a triangle whose vertices are located in distinct segments, and reports the reduction in the RN count relative to the the mst-edge based connection of the corresponding three segments. In the next step, CIST connects groups of three segments with positive gain (reduction in RN count) by steinerizing the corresponding triangle. Finally, the remaining segments are federated via steinerizing mst edges. This phase is completed without any

deployment. The output of CIST is a set of target locations, T , and the set of interface nodes, I in each partition. These are used in Phase II as explained next. An illustration of how CIST works is given in Figure 4.1.

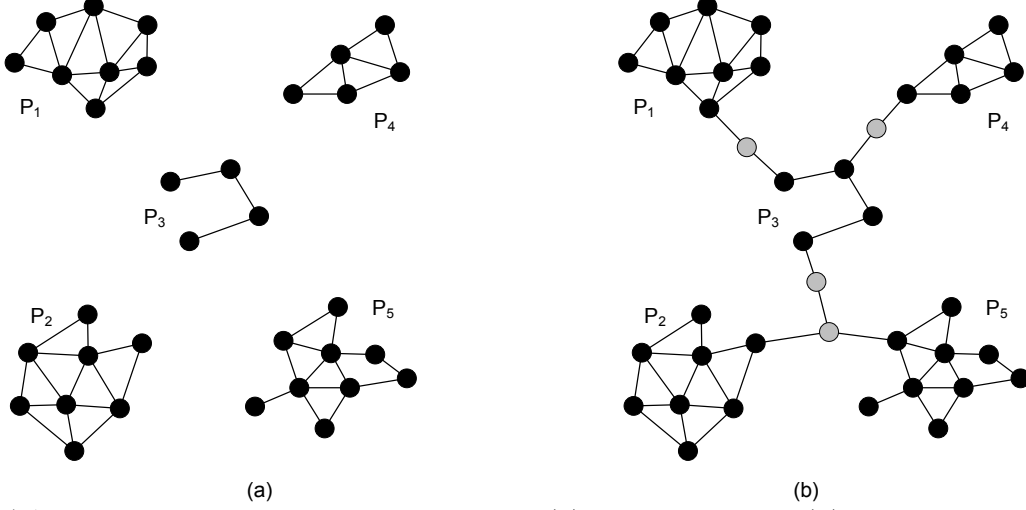


Figure 4.1: An illustration of how CIST works. (a) Initial topology (b) Resulting topology of CIST where the location of gray nodes (RNs) will be used as target locations in Phase II.

4.2.3 Phase II - Connecting Partitions

In Phase II, partitions are connected by moving the existing nodes to the positions determined in Phase I. Therefore, our problem is now reduced to minimizing the total movement distance when the locations in set T are occupied by a certain number of nodes (which is equal $|T|$) from the given partitions. Two sub-problems can be identified here: First, selecting the nodes to be relocated without violating the inter-partition connectivity. Second, deciding the locations to be filled in T by the selected nodes. The main goal for these sub-problems is reducing the total distance to be traveled at the end. We now explain how each sub-problem is addressed.

4.2.3.1 Connectivity Maintenance

In order to maintain connectivity, we determine Connected Dominating Set (CDS) of the partition and pick dominatee nodes as potential nodes for movement. Dai's algorithm

[44] is used to determine the CDS of the partition. Once the CDS is determined, each node can check whether it is a dominator (i.e., an element of the CDS) or a dominee (i.e., can reach to a dominator within one-hop) as shown in Figure 4.2. The set of dominee nodes in a partition Par_i is referred to as D^i . Note that if a dominee node leaves the partition, the connectivity of the partition is still maintained and thus no further topology adjustments are needed within the partition.

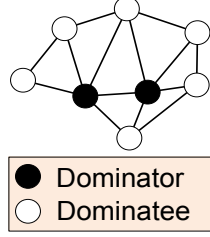


Figure 4.2: CDS consisting of dominator nodes in a partition.

The main issue regarding the dominee node selection is that multiple dominees may exist in $D = \bigcup_{i=1}^k D^i$ which can be picked for relocation to a certain location in T . This is addressed in the next sub-problem where we decide which dominee node should be picked from D for filling a particular location in T .

4.2.3.2 Determining Specific Locations for Dominees

We follow a greedy heuristic for the second sub-problem by considering the distances between the target locations and the dominee nodes. Specifically, a target location in T is occupied with an available dominee node from D which has the minimum distance to that location. However, a dominee node can be the closest node for multiple target locations. Therefore, we consider all the target location-dominee node pairs (T_i, D_i) and pick the node with the minimum distance to its corresponding target locations.

It is important to note that once the dominee nodes start to leave, this may require a change in the existing CDS of a partition. For instance, some of the dominator nodes may become dominees and will be eligible to be selected in the rest of the process. Therefore, the CDS should be continuously updated after each movement as seen in Fig. 4.3b. With

this update, the approach still works even if the initial number of dominatee nodes, $|D|$, is less than the number of target locations, $|T|$. In addition to the CDS update, the T_i - D_i pairs are also updated after each movement so that a new minimum distance is picked next. Algorithm stops when the network connectivity is restored (i.e., all target locations are filled). An illustration of the algorithm is depicted in Fig. 4.3. In this figure, the initial partitioned topology is given with possible target locations. In Fig. 4.3b, the connectivity is restored by moving two dominatee nodes to the target locations.

Note that dominatee nodes are not allowed to move under certain conditions. For instance, when a dominatee node is also an interface node, then it should not be moved to avoid further partitioning. However, if the interface node is the only node (left) in its partition, then it can move as long as it does not serve as a *bridge* node among multiple partitions. A *bridge* node is defined as a node which has more than 1 target locations within its transmission range. Since CIST assumes that interface nodes are the connection points, it places the least number of RNs that will get connected with the interface nodes. Therefore, if an interface node will be connecting multiple partitions, it cannot be moved by our algorithm.

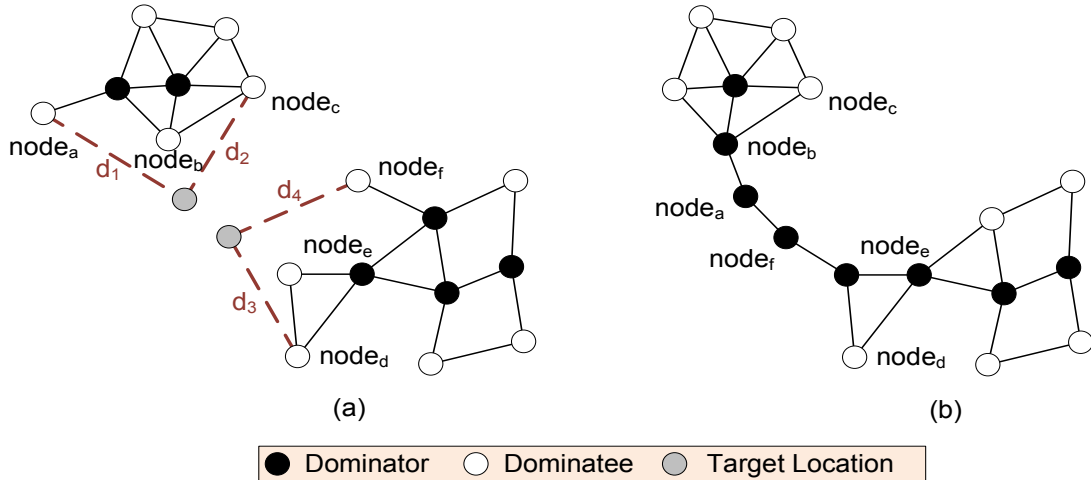


Figure 4.3: Illustration of Phase II. (a) Initial topology. (b) Final topology. Note that $d_1 < d_2$ and $d_4 < d_3$. Interface nodes $node_b$ and $node_e$ are not allowed to move. Note the increase in the number of dominatees after CDS update.

4.2.4 Algorithm Pseudo Code and Analysis

The pseudo-code for the proposed heuristic is given in Algorithm 1. First, the set of target locations, T , are obtained from CIST algorithm in line 2 as mentioned in Section 4.2.2. Afterwards, all of the dominatee nodes in all of the partitions (referred to as set D) are determined through CDS in line 4. Subsequently, the elements in T and D are considered one-by-one and the D_i-T_j pair with the minimum distance is selected for movement (lines 5-13). Finally, the dominatee node D_i moves to the location of T_j in line 17. This operation is performed until reaching connectivity and CDS is updated each time. As mentioned in Section 4.2.3, there are certain conditions to allow a dominatee node to move. These conditions are checked in line 10 in $\text{canMove}(\text{node}_j)$ function. The pseudo-code for the function $\text{canMove}(\text{node}_j)$ is given in Algorithm 2. Basically, this function first checks whether node_j is a member of I and it is the only element in its partition (lines 1-4). If node_j is the only element, then it can move as long as it is not a bridge node which is checked in lines 5-12.

4.2.4.1 Time Complexity Analysis

For a network with n nodes and k partitions, the cost of the CIST algorithm (Phase I) can be computed as $O(((n \log n) + Yk^2))$ based on the pseudo-code given in [43] where Y is the cost involved in SMT creation. CDS can be found in polynomial time [44]. $D_i - T_j$ pair with the minimum distance can be found with a cost of $O(mt)$ for m dominatee nodes and t target locations. However, the movement check adds another cost of $O(t)$ and the total cost for Phase II becomes $O(mt^2)$. Thus, computational complexity of the algorithm can be expressed as $O(((n \log n) + k^2) + mt^2)$ or simply as $O(n \log n)$ assuming $m < n \log n$ and $t^2 < n$. Note that when $n = k$ (i.e., there are n partitions with 1 node), then we can expect $t \geq n$ and the complexity would be $O(n^2) + mt^2$ or simply $O(mt^2)$.

Algorithm 1 connect()

```
1: partitions =  $Par_1, Par_2, \dots, Par_k$ 
2: partitions =  $Par_1, Par_2, \dots, Par_k$ 
3: for  $i = \{1, 2, 3, \dots, k\}$  do
4:   interfacePntsi = findBoundaryNodes( $Par_i$ )
5: end for
6: for  $i = \{1, 2, 3, \dots, k\}$  do
7:   for  $j = \{1, 2, 3, \dots, k\}$  do
8:     MST_Edgesi,j = runMST(interfacePntsi, interfacePntsj)
9:   end for
10: end for
11: for  $i = \{1, 2, 3, \dots, k\}$  do
12:   for  $j = \{1, 2, 3, \dots, k\}$  do
13:     for  $\delta = \{1, 2, 3, \dots, k\}$  do
14:       Targets = findMin(MST_Edgesi,j, MST_Edgesi,\delta, MST_Edgesj,\delta)
15:     end for
16:   end for
17: end for
18:  $T = CIST()$ 
19: while length(partitions) > 1 do
20:    $D = \bigcup_{i=1}^k D^i$ 
21:   minDist =  $\infty$ 
22:    $m = \text{length}(\bigcup_{i=1}^k D^i)$ 
23:    $t = \text{length}(T)$ 
24:   for  $i = \{1, 2, 3, \dots, m\}$  do
25:     for  $j = \{1, 2, 3, \dots, t\}$  do
26:       if distance( $D_i, T_j$ ) < minDist and canMove( $D_i$ ) then
27:         minDist = distance( $D_i, T_j$ )
28:         minD =  $D_i$ 
29:         minT =  $T_j$ 
30:       end if
31:     end for
32:   end for
33:   relocate minD to minT
34:   update partitions
35: end while
```

Algorithm 2 canMove($node_j$)

```
1: if  $node_j \in I$  and  $length(Par_c) > 1$ ,  $node_j \in Par_c$  then
2:   return false
3: end if
4: if  $node_j \in I$  and  $length(Par_c) = 1$ ,  $node_j \in Par_c$  then
5:    $t = length(T)$ 
6:    $neighborcount = 0$ 
7:   for  $i = \{1, 2, 3, \dots, t\}$  do
8:     if  $distance(node_j, T_i) \leq r$  then  $neighborcount++$ 
9:   end if
10:  end for
11:  if  $neighborcount > 1$  then
12:    return false
13:  end if
14: end if
15: return true
```

4.2.4.2 Total Movement Complexity Analysis

In terms of total movement distance, the worst case occurs when there are k partitions with line topologies placed in a row. Assuming that $t = |T|$, there needs to be t total movement for connectivity establishment. In each movement a different distance is traveled by the nodes. While we cannot know individual distances, we can determine the maximum travel distance traveled and thus find an upper bound in the total movement distance.

Theorem 4.2.1. *The worst case movement complexity is $O(nrt/2)$.*

Proof. When multiple partitions start connecting in a row, eventually there will be two partitions left to be connected. In the worst case, the number of nodes in these two partitions will be equal (i.e., $\frac{n}{2}$) and they will be maximally placed apart assuming a transmission range of r . As a result, the distance from a dominee node (which will be at the end of the line topology) to the closest target location will be $\frac{nr}{2}$ as seen in Figure 4.5. If each dominee node is assumed to move this distance, then we can get an upper bound on the total movement distance which is $O(\frac{nr}{2}t)$. $\square$

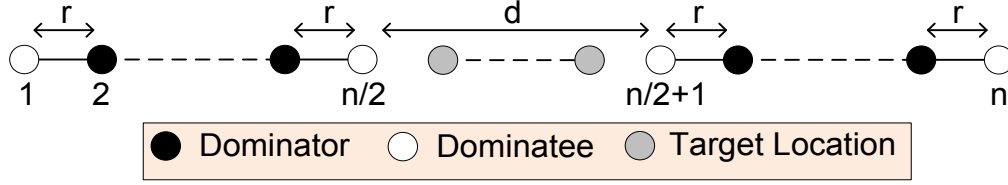


Figure 4.4: Worst case total movement before two partitions get connected. The distance from $node_1$ to the closest target location is $\frac{nr}{2}$

4.3 EXPERIMENTAL EVALUATION

The effectiveness of the proposed approach is validated via simulation. Connected topologies are considered in a 800×800 m area. The transmission range is set to 50m and the number of nodes and partitions are varied. To form the desired number of partitions, cut-vertex nodes (i.e., nodes whose absence partitions the network) in the topologies are selected randomly to be failed. Each test case has been tested with 30 different topologies for significance and the average is reported.

Two performance metrics are considered. Total movement distance is picked for assessing the energy burden of the approach given that movement is the dominant factor in energy consumption. In addition, to evaluate the completion time of the recovery, we picked the maximum movement distance of a node in the network.

We picked two baselines to compare with the proposed approach. The first baseline is the near-optimal solution presented in [16]. Providing the best solution known so far, this solution is used as a benchmark to check the performance of the proposed approach. However, since [16] can handle at most 30 nodes, we also used a second baseline to assess the scalability of the approach. The second baseline is a variation of the approach proposed in [26] where the failed node locations are used to reposition the nodes. While [26] additionally considers coverage, we only considered connectivity to be able to compare with our approach. In the graphs, the optimization-based approach in [16] is represented as *OBA*, our relay-based proposed approach is represented as *RBA* and the failed-node based Approach in [26] is represented as *FLA*.

4.3.1 Performance Comparison with the Near-optimal Solution (OBA)

The proposed approach is first compared with OBA by varying the number of nodes from 10 to 30 and the number of partitions from 2 to 5.

4.3.1.1 Total Movement Distance

The results for total movement distance metric are depicted in Figure 4.5 and Figure 4.6 respectively. The results in Figure 4.5 show that for both approaches, total movement distance falls when the number of partitions is fixed and the number of nodes is increased in the network. This is expected due to the increased probability of having closer partitions in the neighborhood. The results also indicate that our approach, RBA, performs very close to OBA. The performance gap between two approaches declines with the increasing number of nodes and becomes almost negligible when $n = 30$. The slight gap can be attributed to the non-optimality in the target location selection and greedy-based movement. However, this slight advantage of OBA in the movement is at the expense of higher time complexity which eventually causes the approach not to produce any solutions when the number of nodes is beyond 30.

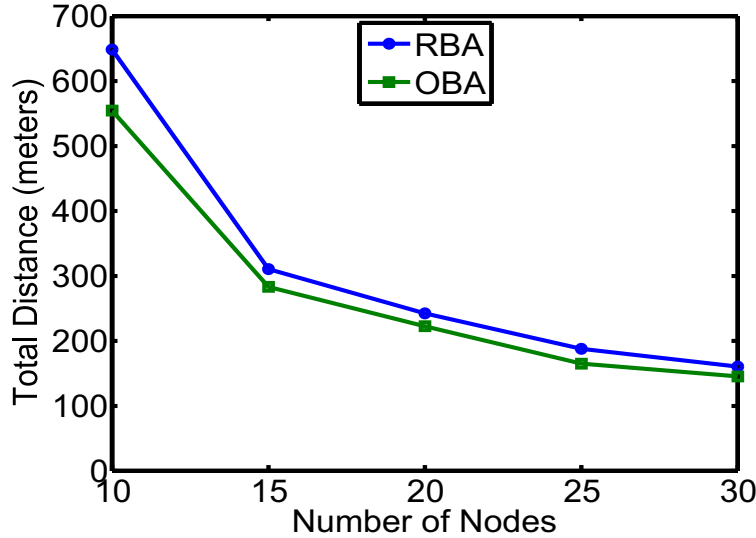


Figure 4.5: Total travel distance with 3 partitions and varying number of nodes.

When the number of nodes is fixed and the number of partitions is increased, the

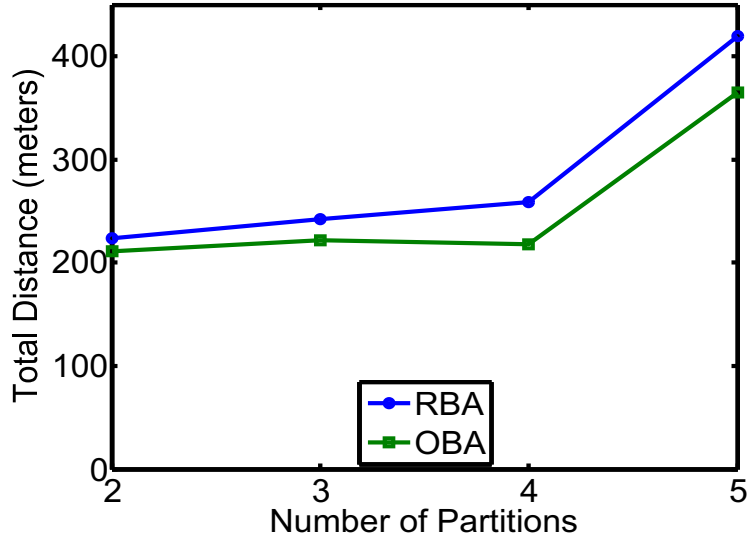


Figure 4.6: Total travel distance with 20 nodes and varying number of partitions.

performances of both approaches are still very close as seen in Figure 4.6 despite a slight increase in the gap. The performance gap slightly increases especially until the number of partitions is 3. As expected, the total movement distance increases when the network is partitioned into more pieces due to the increased recovery cost for each partition. This is attributed to the increased number of possible locations that can be used to move the nodes. As shown in [43], the number of relay nodes needed to connect the partitions increases when the number of partitions is increased. Such increase indicates that more target locations will be available and minimizing the movement cost with a greedy approach may not always give the best solution. Such problem can be better handled with OBA albeit with a high complexity cost. Nonetheless, the gap does not increase further and stay almost constant when the number of partitions is 5.

4.3.1.2 Recovery Delay

We also evaluated the movement recovery delay required to guarantee a solution. Maximum distance traveled by any node is considered as the recovery delay assuming 1 meter can be traveled in 1 second by a node [3]. The results for increasing number of nodes

and partitions are given in Figure 4.7 and Figure 4.8 respectively.

It can be observed in Figure 4.7 that as in the case of total travel distance (and due to same reasons), our approach, RBA, performs very close to OBA. The recovery delay is decreasing for both approaches when the number of nodes is increased. This is expected due to the increased possibility of having closer partitions in denser networks.

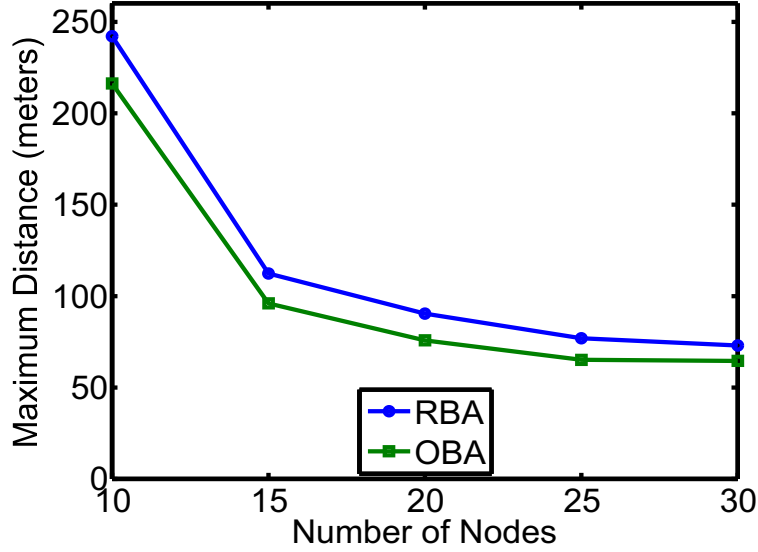


Figure 4.7: Maximum travel distance with 3 partitions and varying number of nodes.

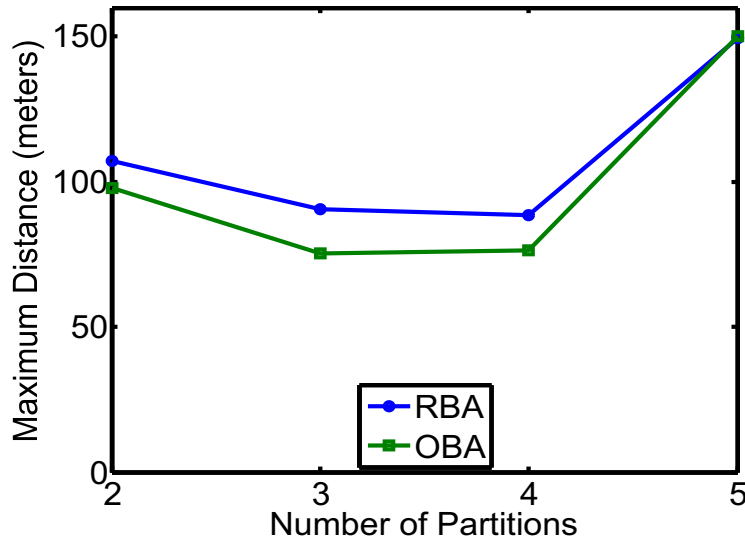


Figure 4.8: Maximum travel distance with 20 nodes and varying number of partitions.

When we look at the performance with increased number of partitions, we observed that RBA and OBA have similar recovery delay patterns as seen in Figure 4.8. However, the

recovery delay is independent from the number of partitions as it depends on the distances of the nodes in any partition to the target locations in the damaged area. Therefore, a constant performance gap among both approaches is not expected.

4.3.2 Performance Comparison for Assessing Scalability

To assess the scalability of the proposed approach, it is compared with FLA in [26].

4.3.2.1 Total Movement Distance

The results for the total movement distance are given for varying number of nodes and partitions in Figure 4.9 and Figure 4.10 respectively.

It can be observed from Figure 4.9 that the FLA performs significantly worse than RBA especially when the number of nodes is minimum in the network. Even though the performance gap declines when the number of nodes is increased to 100, it remains constant afterwards. This result is attributed to the fact that potential locations for the nodes to move are reduced significantly. Since FLA considers only the failed node locations, such locations do not necessarily indicate the minimum set to guarantee connectivity. Therefore, more movements will be made in FLA which eventually increases the total movement cost.

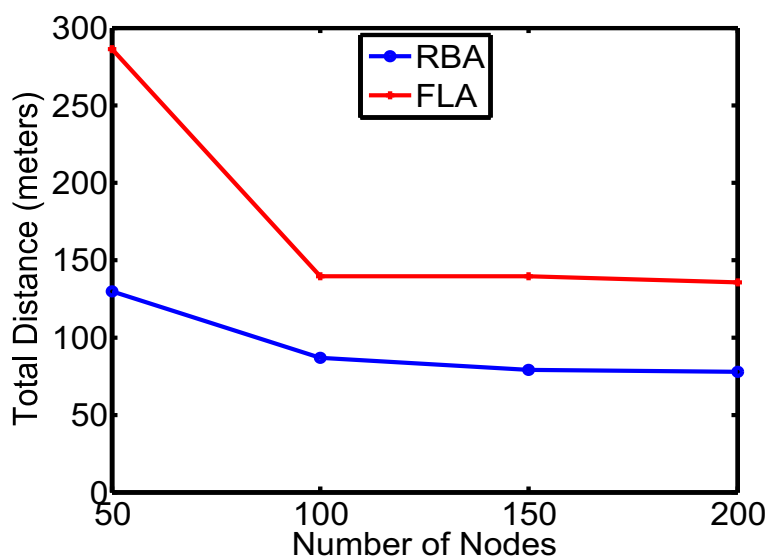


Figure 4.9: Total travel distance with 2 partitions and varying number of nodes.

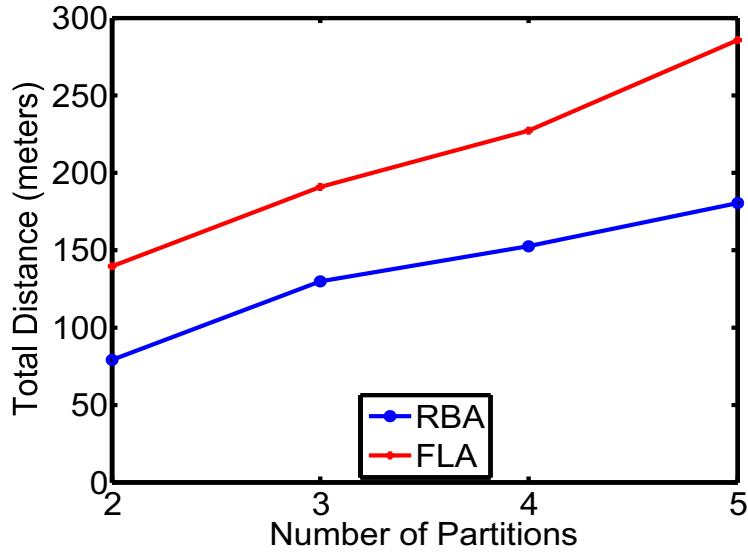


Figure 4.10: Total travel distance with 150 nodes and varying number of partitions.

Figure 4.10 portrays how both approaches react when the number of partitions is increased. Cost increase for both approaches is expected due to the increased need for recovery at different locations. The results show that RBA outperforms FLA and even increases the performance gap further with the increased number of partitions. This can be attributed to the increased number of failed node locations as opposed to target locations when more partitions are considered.

4.3.2.2 Recovery Delay

The results for recovery delay are given for increasing number of nodes and partitions in Figure 4.11 and Figure 4.12 respectively. It can be seen from Figure 4.11 that our approach outperforms FLA when the number of nodes is increased. Again, this is attributed to the number of target locations and their spatial distribution which help minimize the maximum distance traveled for all the nodes.

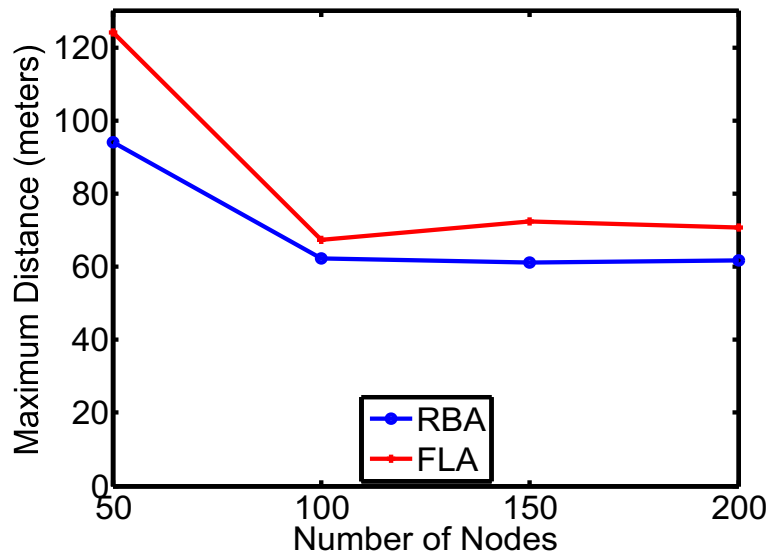


Figure 4.11: Maximum travel distance with 2 partitions and varying number of nodes.

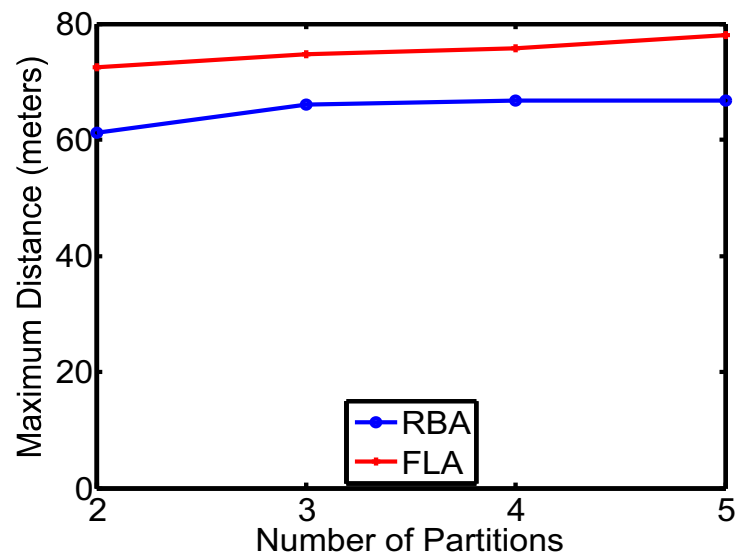


Figure 4.12: Maximum travel distance with 150 nodes and varying number of partitions.

For varying number of partitions, as is the case in the previous subsection, there is no correlation between the recovery delay and the number of partitions. Nonetheless, for any number of partitions, RBA again outperforms FLA.

4.4 SUMMARY

In this chapter, we have proposed an effective yet scalable connectivity restoration heuristic which guarantees connectivity in MSNs or WSNs. The idea is to reduce the set of locations for the existing mobile nodes in the network to the locations of relay nodes that would ensure connectivity with the least count. A greedy-based heuristic is followed to move the nodes in the partitions whose absence would not violate the inter-partition connectivity. This was determined by using a CDS.

The evaluation of the proposed approach indicated its effectiveness with respect to the near-optimal solution, OBA, by providing very close results in terms of total movement distance and recovery delay metrics. Yet, the approach is shown to provide results when the network is scaled which was not possible with OBA. At such scales, the proposed approach outperformed another existing approach where failed node locations were used to restore connectivity.

CHAPTER 5

DISTRIBUTED CONNECTIVITY RESTORATION IN MOBILE SENSOR NETWORKS USING GAME THEORY

In this chapter, we address the problem of minimizing the movement cost of connectivity restoration in a partitioned network using existing mobile nodes. The approach we propose handles recovery in a semi-distributed manner assuming complete information about the post-failure network topology may not always be available in a realistic scenario. Compared to centralized heuristics, this approach is self-healing and it assures recovery with uncertain network/failure information. Game Theory is employed to improve the decision making process for the selection of the target partitions to be connected with incomplete information. Using a probabilistic model, the nodes within each partition collaborate to estimate a Nash equilibrium for other partitions and compare it with the equilibrium of its own partition to decide for a movement and the direction of a possible movement. The proposed Game Theory based distributed approach is evaluated through extensive simulations. The results have revealed that the movement cost can be minimized at both the network level and the individual node level while maximizing the network coverage by applying the proposed approach compared to the centralized heuristics. The approach also scales well and performs very close to the near-optimal solution given in Chapter 3 and unlike the near-optimal solution it does not require complex computations while guaranteeing a solution.

5.1 PRELIMINARIES

5.1.1 Assumptions

We assume a connected WSN where the nodes are randomly deployed with a single sink node which collects the data from the network. The sensors and the sink are assumed to be able to move on demand [45] and have limited transmission range so that the nodes

collaborate in order to relay data to the sink. We assume a massive damage causing simultaneous node failures creating multiple disjoint partitions in the network. The damage is assumed to be at an arbitrary time but at least late enough to let nodes to determine routing path to the sink node. We assume that the nodes have enough power to complete their movement and they can locate themselves using one of the existing approaches [46]. The nodes are also assumed to be equipped with visual sensors/cameras to identify the existence of other partitions for deciding their movement direction. Note that such a camera already exists in mobile sensor robots today [45]. The field of view (FoV) and depth of field (DoF) in these cameras refer to the angle and the maximum distance the camera can accurately capture an image respectively. DoF can be adjusted based on camera lens parameters and can go up to infinity [47]. Therefore, a mobile sensor's camera can span 360° with movement and detect the existence of other sensors within its distance.

5.1.2 Problem Definition

A connected network is given, composed of m nodes, such that $\forall Node_i$ is accessible by $\forall Node_j$ for $1 \leq i \leq m$ and $1 \leq j \leq m$ either directly or through other nodes. At some arbitrary moment, $1 \leq f \leq m - 2$ nodes are removed from this network creating $k > 1$ partitions. Our problem can be stated as: "Given a set of n sensors along with their locations in k partitions where $n = m - f$, provide a solution which integrates k partitions into a single connected network comprising n sensors by relocating the sensor nodes and the sink while minimizing $\sum_{i=1}^n d_i$ where d_i is the total travel distance of $node_i$ ".

5.2 CENTRALIZED HEURISTIC APPROACHES

Before providing the details of the proposed approach, we define two centralized heuristic approaches which we compare with the proposed approach. The centralized heuristics also strive to minimize total movement cost while restoring connectivity in a partitioned network. Note that these will be used as baselines and thus the necessary information is

assumed to be available theoretically.

5.2.1 Density Based Centralized Heuristic (DeCH)

The first approach, namely Density Based Centralized Heuristic, claims that the total movement cost can be minimized if the biggest partition remains stationary and the other partitions performs movement. Thus, the partitions will be reunited around the biggest partition. To achieve this, the biggest partition is identified first. Then, each of its nodes are added, without any movement, to the new set Par_{final} which eventually will be the resulting topology at the end. Other partitions are sorted based on the number of nodes they have and the partition with the most number of nodes, let it be Par_j , is considered first (i.e., density based heuristic). The nodes of Par_j are inspected one by one with the nodes in Par_{final} . The order of the nodes to be inspected is based on their distance to the center of the topology, a heuristic decision which aims to minimize overall movement. If the inspected node in Par_j , let it be $node_{current}$, is within the transmission range of the closest node to itself in Par_{final} , let it be $node_{min}$, this node is added to the new set without movement. Otherwise it is relocated to a point that would minimize its movement distance but still be in the transmission range of the new set. This new position is determined by a formula which considers transmission range (r), distance between $node_{current}$ and $node_{min}$ (d), and the slope of the line formed between $node_{current}$ and $node_{min}$ (α , converted to angle). The new position's x and y coordinates can be expressed as :

$$x = x_{node_{current}} + (d - r) * \cos(\alpha) \quad (5.1)$$

$$y = y_{node_{current}} + (d - r) * \sin(\alpha) \quad (5.2)$$

After relocation, this node is added to Par_{final} . When all of the nodes in Par_j are

added to Par_{final} , either with or without relocation, next partition with the most number of nodes is considered. Algorithm terminates when all of the nodes in all the partitions are considered and added to Par_{final} . At the end Par_{final} represents the final topology and the total distance between the locations of the corresponding nodes in the initial topology and Par_{final} is regarded as the total movement cost. The pseudo code for the algorithm is given in Alg. 3.

Algorithm 3 density_based_heuristic(*partitions* : $Par_1, Par_2, \dots, Par_k$)

```

1: Sort partitions =  $Par_1, Par_2, \dots, Par_k$  in descending order based on the partition size
2: Add nodes in  $Par_1$  to  $Par_{final}$ 
3:  $CP$  = Center of network
4: for  $i = \{2, 3, \dots, k\}$  do
5:    $p = Par_i, s = size(p)$ 
6:   Sort nodes  $\in p$  in ascending order based on the distance to  $CP$ 
7:   for  $j = \{1, 2, \dots, s\}$  do
8:     find  $node_{min} \in Par_{final}$  such that  $distance_{node_j - node_{min}}$  is minimized
9:      $minDist = distance_{node_j - node_{min}}$ 
10:    if  $minDist < r$  then
11:      add  $node_j$  to  $Par_{final}$ 
12:    else
13:       $newX = node_{j_x} + (minDist - r) * (node_{min_x} - node_{j_x}) / minDist$ 
14:       $newY = node_{j_y} + (minDist - r) * (node_{min_y} - node_{j_y}) / minDist$ 
15:       $cost = dist_{node_j - \{newX, newY\}}$ 
16:       $node_{j_x} = newX$ 
17:       $node_{j_y} = newY$ 
18:       $totalCost += cost$ 
19:      add  $node_j$  to  $Par_{final}$ 
20:       $s = size(p)$ 
21:    end if
22:  end for
23: end for

```

Complexity: For a network with k partitions, the cost of sorting the partitions is $\mathcal{O}(k \log k)$. The cost to determine the center of the network is $\mathcal{O}(n)$ for n nodes in the network. Nodes are sorted in each partition which requires an additional $\mathcal{O}(s \log s)$ for s nodes in a partition. For s nodes, the node with the minimum distance to partition is selected among the nodes in Par_{final} . If there are s' nodes in Par_{final} , the cost is $\mathcal{O}(s * s')$. Since there are k partitions, total cost is $\mathcal{O}(n * (k \log k + s * s'))$. With the initial partition sorting and the

central point determination, overall complexity is $\mathcal{O}(k \log k + n + k * (s \log s + s * s'))$. k can be ignored for small number of partitions and since $(s + s') \leq n$, $s * s'$ can be $n/2 * n/2$ in the worst case which $\mathcal{O}(n^2)$. Assuming $s * s' \geq s \log s$, the worst case computational complexity of the algorithm can be expressed as $\mathcal{O}(n + n^2)$ and eventually as $\mathcal{O}(n^2)$ for n nodes in the network.

5.2.2 Distance Based Centralized Heuristic (DiCH)

DiCH also pursues concentrating the partitions around the partition with the most number of nodes. However, this heuristic considers other partitions with a different ordering procedure. Once Par_{final} which represents the final topology is initiated by the nodes of the biggest partition without any movement, other partitions are sorted based on their minimum distance to Par_{final} (i.e., distance-based heuristic). This minimum distance is the distance between the closest nodes of both partitions. Let Par_j be such a partition that has the minimum distance to Par_{final} . The nodes in Par_j are inspected one by one with the nodes in Par_{final} . If the node is within the transmission range of the any of the nodes in Par_{final} , it is added to Par_{final} without movement. Otherwise, it is relocated to a point that would minimize its movement distance but still be in the transmission range of Par_{final} . As also discussed in 5.2.1 this new position can be determined using Equations 5.1 and 5.2 for a node $node_{current}$ in Par_j which determines $node_{min}$ as the closest node to itself in Par_{final} . After relocation, this node is added to Par_{final} . The order of the nodes to be inspected is based on the distance to the closest node in Par_{final} . Upon a node is added from Par_j to Par_{final} , the remaining nodes in Par_j are sorted again based on their distances to the updated Par_{final} .

When all of the nodes in Par_j are added to Par_{final} , the remaining partitions are sorted once again based on their distance to the updated Par_{final} . The partition with the minimum distance to Par_{final} is selected for consideration like before and this process recurses itself until all the nodes of all the partitions are added to Par_{final} . Par_{final} , at

the end, represents the final topology and the total distance between the locations of the corresponding nodes in the initial topology and Par_{final} is regarded as the total movement cost. The pseudo code for the algorithm is given in Alg. 4.

Algorithm 4 distance_based_heuristic($partitions : Par_1, Par_2, \dots, Par_k$)

```

1: Sort  $partitions = Par_1, Par_2, \dots, Par_k$  in descending order based on the partition size
2: Add nodes in  $Par_1$  to  $Par_{final}$ 
3: Sort  $partitions = Par_2, Par_3, \dots, Par_k$  in ascending order based on the minimum
   distance to  $Par_{final}$ 
4: for  $i = \{2, 3, \dots, k\}$  do
5:    $p = Par_i, s = size(p)$ 
6:   for  $j = \{1, 2, \dots, s\}$  do
7:     find  $node_{min} \in Par_{final}$  such that  $distance_{p-node_{min}}$  is minimized
8:     Sort nodes  $\in p$  in ascending order based on the distance to  $node_{min}$ 
9:      $minDist = distance_{node_1-node_{min}}$ 
10:    if  $minDist < r$  then
11:      add  $node_j$  to  $P_{final}$ 
12:    else
13:       $newX = node_{j_x} + (minDist - r) * (node_{min_x} - node_{j_x}) / minDist$ 
14:       $newY = node_{j_y} + (minDist - r) * (node_{min_y} - node_{j_y}) / minDist$ 
15:       $cost = dist_{node_j - \{newX, newY\}}$ 
16:       $node_{j_x} = newX$ 
17:       $node_{j_y} = newY$ 
18:       $totalCost += cost$ 
19:      add  $node_j$  to  $Par_{final}$ 
20:       $s = size(p)$ 
21:    end if
22:  end for
23: end for

```

Complexity: For a network with k partitions, the cost of sorting the partitions is $\mathcal{O}(k \log k)$. Minimum distance between the partitions can be determined with a cost of $s * s'$ which is $\mathcal{O}(n^2)$ for a network with n nodes in the worst case. Assuming s' nodes in Par_{final} , for a partition with s nodes, cost is $\mathcal{O}(s * (s' + s \log s))$; s' to find the closest node in Par_{final} and $s \log s$ to sort the nodes in the partition. Thus, for k partitions, cost is $\mathcal{O}(k * (s * (s' + s \log s)))$. With the initial cost of sorting and minimum distance calculation for partitions, overall cost is $\mathcal{O}(k \log k + n^2 + k * (s * (s' + s \log s)))$. k can be ignored for small number of partitions and since $(s + s') \leq n$, the worst case computational complexity

of the algorithm can be expressed as $\mathcal{O}(n^2 + n^2 \log n)$ and eventually as $\mathcal{O}(n^2 \log n)$ for n nodes in the network.

5.3 GAME THEORY BASED DISTRIBUTED HEURISTIC (GTDH)

5.3.1 Game Theory Background

There has recently been a vast array of studies in which Game Theory is used for solving both the cooperative and non-cooperative based computer science problems in many contexts. In cooperative games, there are agents/players which tend to act together to meet the best interest of the environment or the network they belong to in order to maximize their payoffs. Under cooperative games, strategic complementary is the type of action or decision where nodes within each network are the players encouraging each other in terms of their strategies. In non-cooperative games, there are players and strategies related to the selfish actions of the agents in the same environment. Strategic substitute is the type of action that is related to players who balance one another's action depending on the available strategies by the players.

Another issue is the characteristic of the information in a game, i.e., complete or incomplete information. The incomplete information implies that each player in a network has only the information regarding itself but does not have any information about its neighborhood. In the case of complete information, every node knows the network state. Based on the player/agents' best responses in a game, the equilibrium basically consists of the payoffs and the action sets of the players so that no player will have any incentive to deviate. Payoff refers to the gain of a node in a game obtained as a result of its actions. Once every player in the network best responds in this setting, the network game is assumed to be in equilibrium.

The details on how the model is constructed and how the structure works are given in the next subsection.

5.3.2 Model

5.3.2.1 Overview and Notation

We propose a solution using a Game theoretical approach based on network games [48]. Changes in the network have an impact on the equilibrium and payoffs for games. Normally, strategic complements or strategic substitutes can be used in the game. While the former refers to the situations where players acts in the same manner towards a certain goal whereas the later refers to the cases where players can have conflicting actions. In this work, we basically focus on the network equilibrium in the strategic complementary case where nodes within each network are the players of a cooperative game. The purpose of the game is connecting partitions by changing the degrees of the nodes (i.e., players). The degree of a node is the number of neighbors it has within the network. The used notation throughout this section is given in Table 5.1.

The model consists of a two-stage game with incomplete information in the sense that each node in a partition knows only its degree but does not have any information about its neighbors' degrees. In the model, the network games of complements is defined by $\{Par, A, \{v_k\}_k, P\}$. We first assume that there are finite nodes in each partition such as $Par = \{1, 2, \dots, s\}$. A represents the action set given by $A = \{0, 1\}$ and it is a compact subset of $[0, 1]$. Elements of set A , 0 and 1, for a node, basically represent no relocation and relocation, respectively. In its general form, the action set A is given by $\{a_1, a_2, \dots, a_s\}$. The network which gives the connection between nodes is denoted by a matrix $g \in \{0, 1\}^{s \times s}$, where we assume $g_{node_i, node_i} = 0$ for every $node_i \in Par$ and $g_{node_i, node_j} = 1$ represents that the behavior of $node_j$ in terms of its strategy has an impact on the payoff of $node_i$. The (net) payoff can be defined as what a node gains by choosing a specific action minus the cost of the same action to it after each set of action in a general sense. For example, the cost of choosing action 0 is usually assumed to be zero. This specification helps us to define the set of neighbors for each node and then its degree afterwards.

$N_{node_i} = \{j | g_{node_i, node_j} = 1\}$ gives us the set of neighbors of $node_i$ and the degree of

Table 5.1: Notation for the Proposed Approach

<i>Acronym</i>	<i>Explanation</i>
$node_i$	Sensor Node i
Par_i	Partition i
m	# Partitions in the network
n	# of nodes in the network
A	Action set of nodes
g	Connection Matrix for the network
N_{node_i}	Neighbors of $node_i$
k_{node_i}	Degree of $node_i$
$K_{N_{node_i}}$	Total degrees of all the neighbors of $node_i$
$v_{k_{node_i}}$	Payoff function for $node_i$ with a degree k_{node_i}
b_i	# of boundary nodes in Par_i
b'_i	# of boundary nodes in Par_i which do not intersect with any partition
b_{ij}	# of boundary nodes in Par_i which intersect with Par_j
e_{Par_i}	Equilibrium of Par_i
$P(K_{Par_i})$	The pdf of Par_i based on node degree as the random variable

$node_i$ is denoted by k_{node_i} , where $k_{node_i} = |N_{node_i}|$ and shows the connections of $node_i$. Moreover, $v_{k_{node_i}}(A_{node_i}, A_{N_{node_i}})$ defines the payoff of $node_i$ if its profile of actions is given by $A = \{a_1, a_2, \dots, a_s\}$. $v_{k_{node_i}}$ is the payoff function for $node_i$ with a degree k_{node_i} . Two nodes with the same degree have the same payoff function. It also varies accordingly with the node degree. The calculation of such payoff function is determined by the connections that a node has. Hence, the payoff function for $node_i$ depends on its action and actions of its neighbors and every neighbor satisfies assumption 1 below. Finally, the probability distribution P , explicitly represented by $P(K_{N_{node_i}}|k_{node_i})$, where $K_{N_{node_i}}$ is the degrees of the neighbors of $node_i$. This conditional probabilistic distribution defines the information structure used throughout the model.

We also use two key assumptions indicated by [48] besides strategic complementarity property of the nodes in a network :

- (a) $v_{k+1}(a_{node_i}, (a, 0)) = v_k(a_{node_i}, a)$ for any $(a_{node_i}, a) \in A^{k+1}$. A^{k+1} consists of the possible elements of the action set with $k + 1$ -dimensional actions vector. This property mainly implies that for a node having no additional neighbor, it is assumed that it

will have the same payoff if another node whose action choice 0 decides to connect to it.

- (b) The probability distribution P exhibits positive neighbor affiliation given by if for all $k' > k$ and any non-decreasing function, $f : \{0, 1, \dots, s-1\}^k \rightarrow R$, the following inequality is satisfied: $E_{P(\bullet|k')}[f] \geq E_{P(\bullet|k)}[f]$, where $E_{P(\bullet|k)}[f] = \sum_{K_{N_{node_i}}} P(K_{N_{node_i}} | k_{node_i}) f\{k_{node_1}, k_{node_2}, \dots, k_{node_m}\}$ and $m \leq k_{node_i}$. This assumption helps us to determine the correlation between the degrees of the neighbors in a network.

5.3.2.2 Reaching Equilibrium

In the first stage of the game, we assume each sensor node acts within its partition under strategic complementarity. Without loss of generality, each partition is assumed to have a symmetric *Bayes – Nash* equilibrium, which is defined by its threshold degree with its probability distribution. The threshold degree represents the dominating partition's equilibrium degree. Our main goal is to have them reach the equilibrium level of the partitions which they would like to communicate.

We achieve that in the second stage. Mainly, for two partitions with the probability distributions of P and P' , we know that there will be a non-decreasing equilibrium of a partition with a probability distribution P which dominates a non-decreasing equilibrium of a partition with a probability distribution P' when P dominates P' . Once we find this sort of relation between each partition pairs, each partition will then have the same equilibrium with the partition having a dominant probabilistic distribution compared to its distribution. Here, we note that we continue to assume the strategic complementarity property even among the node-partition relation.

Consequently, we have every partition having the same equilibrium with at least one partition as described above. Then, we are able to compare probability distributions among

the partitions in the second stage of the game. After a similar binary comparison of partitions among themselves, we have a single symmetric *Bayes – Nash* equilibrium of the partition¹ with the dominant probability distribution.

5.3.3 Distributed Heuristic based on Game Theory

5.3.3.1 Approach Overview

In a WSN, when a group of sensors lose connectivity with the sink node due to failures of neighboring nodes, connectivity can be regained by moving the nodes towards damaged areas. However, considering the distributed environment, choosing the best node to move, and determining the optimal movement direction can be challenging. As a result, in each partition, the nodes in the partition should first collectively decide whether they must perform a movement or not. Then, in case of a movement decision, a target partition should be identified if the number of partitions in the network is more than 2. Such a cooperative decision must be made by the nodes in the partition to minimize overall movement cost.

Note that when the presence of other partitions and the movement direction to access such partitions are unknown, movement efforts cannot guarantee connectivity in the end. This is the main reason why the existing approaches in the literature cannot be used. By exploiting primitive cameras on the mobile nodes (e.g., robots or sensors), a node in a partition can roughly recognize the existence of other partitions via the sensors that sit in its FoV. This recognition may help determine the direction for movement of sensors in the partition. We would like to note that this recognition will not provide us the exact number of sensors in a partition since there may be blocking due to sensors as well as other obstacles in the environment. This is not an issue to be concerned with as we only need to gather the direction information and a rough density of sensors in that partition. A Game Theoretic approach is proposed in this chapter to improve the decision making process for

¹A detailed discussion on the existence of a symmetric equilibrium and proofs can be found in [48]

movement.

Given a partition and its neighboring partitions, the first task is to estimate equilibrium for that partition and its neighboring partitions. Once the equilibrium of the other partitions and the own partition (Par_i) are computed, these equilibria are used to determine the probability distributions of the partitions. After this step, partition Par_i repositions its nodes towards partition Par_j for $\exists Par_j$ with the highest probability distribution among m partitions in the network. The nodes are moved one by one until two partitions gets connected. The node for relocation is picked according to the distance of the nodes to the other partition which is assumed to be detected by the cameras.

5.3.3.2 Computing the Equilibrium and Probability Distribution

In this subsection, we explain how a partition determines its own equilibrium/probability distribution as well as the neighboring partitions' equilibrium/probability distribution. We assume that there would be a partition leader which will collect the information, perform this computation and inform the necessary node on behalf of the partition. The details of leader election and messaging among nodes are skipped as the cost of this messaging is not significant compared to movement. Before describing the equilibrium function, we provide some definitions.

Definition. *Boundary Node:* A boundary node is a node which experiences communication failure with its upstream node in its route to the sink. In other words, $node_i$ is a boundary node with a next hop $node_j$ in a partition Par_i such that $node_i \in Par_i$ and $node_j \notin Par_i$.

Definition. *Intersecting Partition:* For a boundary node $node_i$, an intersecting partition is the one which has a node within the FoV of the camera of $node_i$.

Definition. *Probability Distribution Function (pdf):* A special case of the conditional probability defined in Section 5.3.2, our used pdf is based on the node degrees in a partition and is denoted by $P(T)$ where T denotes a random variable. Assuming a uniform distribution

of the node degrees in the partition, T can take values between 0 and q where q is the total number of nodes in the partition. Then $P(T) = \frac{1}{cq}$ where c is a constant that is dependent upon the number of nodes/boundary nodes in a partition and the number of boundary nodes in a neighboring partition.

Computing Equilibrium of a Partition: The equilibrium of a partition is associated with the total degree of the boundary nodes which do not have any intersecting partitions. Let b'_i be the number of such boundary nodes in partition Par_i which do not have any intersecting partitions in the network, and let k_{node_w} represent the degree of $node_w$ as defined before. The equilibrium e_{Par_i} of partition Par_i can formally be stated as follows:

$$e_{Par_i} = \sum_{w=1}^{b'_i} k_{node_w} \quad (5.3)$$

The pdf of a partition is associated with the total number of nodes in the network and the number of the boundary nodes which do not have any intersecting partitions, b'_i . Pdf is also dependent on the equilibrium of the partition. Using the total number of nodes in the network, n and the total node degrees in partition Par_i , K_{Par_i} , pdf of Par_i is defined as follows:

$$P(K_{Par_i}) = \frac{e_{Par_i}}{(b'_i + 1) * n} \quad (5.4)$$

Computing Equilibrium of a Neighboring Partition: The main idea in computing the equilibrium of other partitions in the network is using the former routes of sensors to the sink. The computed equilibrium for other partitions is estimation and is closely related with the degree of boundary nodes intersecting other partitions. Node degree of a boundary node can be regarded as a weight of that boundary node in the partition. Note that if the total

degree of the boundary node increases, more nodes are expected to be connected to the sink node over these nodes. Let b_{ij} be the number of boundary nodes in partition Par_i which intersects with partition Par_j as defined before. For k_{node_w} representing the degree of $node_w$, the equilibrium e_{Par_j} of partition Par_j can formally be stated by using Equation 5.5:

$$e_{Par_j} = \sum_{w=1}^{b_{ij}} k_{node_w} \quad (5.5)$$

The pdf for Par_j can be defined once e_{Par_j} is determined. Total number of boundary nodes in Par_i (i.e., b_i) and the number of boundary nodes intersecting with Par_j (i.e., b_{ij}) are utilized with the equilibrium e_{Par_j} to define the probability distribution. The pdf of Par_j can be defined as follows:

$$P(K_{Par_j}) = \frac{e_{Par_j}}{(b_i - b_{ij} + 1) * n} \quad (5.6)$$

A sample scenario is given in Fig. 5.1 to demonstrate how Par_2 estimates the equilibrium of Par_1 in a network of 200 nodes (i.e., $n = 200$). 7 boundary nodes exist in Par_2 and 5 of them are intersecting with Par_1 . The degrees for the boundary nodes are 4, 3, 4, 3, 4, 3 and 4 respectively from left to right. Therefore, Par_2 calculates its own equilibrium e_{Par_2} as 8 and estimates e_{Par_1} , which is the equilibrium of Par_1 , as 17. Once e_{Par_2} is determined, Par_2 can define its pdf. Since 18 out of 23 nodes exist in Par_2 which is not intersecting with other partitions, $P(K_{Par_2})$ can be determined by $\frac{8}{19*200}=0.0021$. Par_2 can also define a pdf for Par_1 . Since 5 out of 7 boundary nodes intersecting with Par_1 , $P(K_{Par_1})$ can be determined by $\frac{17}{3*200}=0.0283$. Since Par_2 defines a higher pdf for Par_1 than its, it picks Par_1 as the target partition. For more than 2 partitions, maximum pdf is considered for

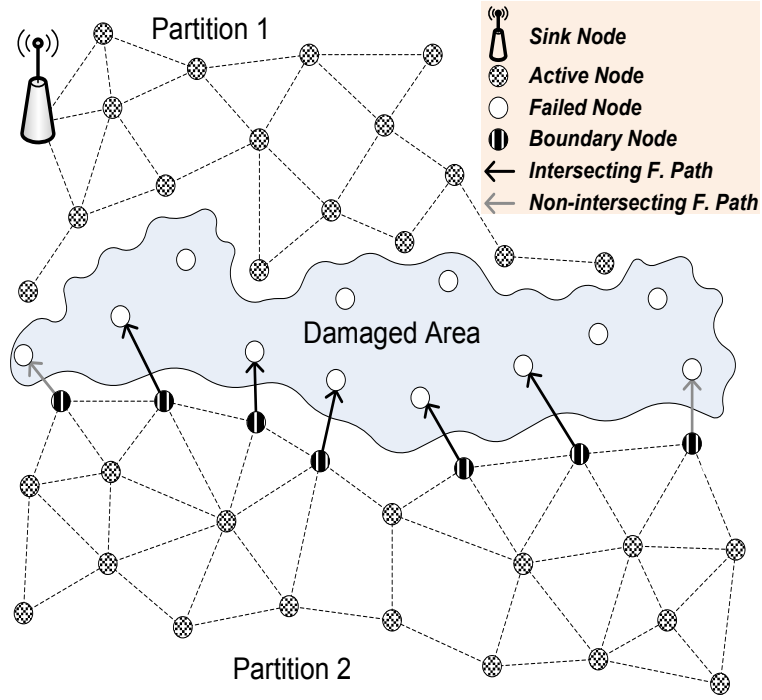


Figure 5.1: Estimating the Equilibrium

comparison. The resultant recovered network is depicted in Fig. 5.2.

5.3.3.3 Algorithm Pseudo Code

Algorithm 5 presents the computation of equilibrium for partition Par_i by a node $node_a$ which has the next hop $node_b$. Note that connectivity of two partitions does not necessarily guarantee the connectivity with the sink node. Therefore, once the partition gets connected to the other partition, the algorithm is run again unless the sink node connectivity is achieved.

Once computed by the individual nodes, equilibria are incorporated to pdfs for each partition. Nodes cooperatively make the movement decision for the partition. The simple pseudo-code in Algorithm 6 describes how the partition reacts to the pdfs for partition Par_i in a network with k partitions.

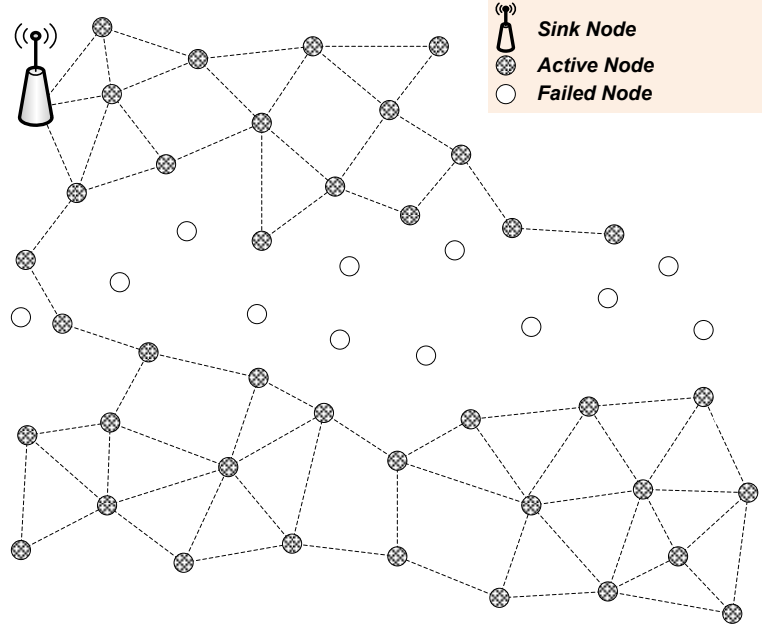


Figure 5.2: Network after recovery

5.4 EXPERIMENTAL EVALUATION

In this section, we start with portraying the experiment setup, then define the performance metrics, followed by the description of the baselines. At the end, we present and discuss the results.

5.4.1 Simulation Setup

Experiments have been carried out to test the efficiency of the proposed approach. We deployed nodes randomly in an application area of 800X800 meters with a transmission range of 40 meters to form a connected network. To form partitions with a given sensor count, cut-vertex nodes are selected randomly to be failed. In the test topologies, when the number of partitions is fixed to 2, number of nodes is varied from 50 to 200; and when the number of nodes is fixed to 150, number of partitions is varied from 2 to 5. For each case, 30 different topologies were created and tested for significance and the average is reported.

Algorithm 5 computeEquilibrium $\{node_a\}$

```
1: if  $node_a \in$  boundary nodes in  $Par_i$  then
2:    $b_i += 1$ 
3:   while  $\exists Par_j$  in visual range,  $i \neq j$  do
4:     if  $line_{node_a-node_b}$  FoV  $Par_j$  then
5:        $b'_j += 1$ 
6:        $K_{Par_j} += k_{node_a}$ 
7:        $\#intersect\_nodes_{Par_i} += 1$ 
8:     else
9:        $K_{Par_i} += k_{node_a}$ 
10:    end if
11:  end while
12: end if
```

5.4.2 Metrics

The performance of the proposed approach has been evaluated based on the following metrics:

- *Total Travel Distance*: This metric measures total distance traveled by all the nodes which perform movement. Total travel distance has a high adverse effect on the battery[5] and dominates the total energy consumption of a node. Therefore, it is crucial to minimize this cost to reduce the total energy consumption in the network. Note that messaging cost is not considered given that it would be negligible compared to node movement cost.
- *Maximum Travel Distance*: This is to assess the maximum distance traveled by any node in the network. Maximum travel distance implies the recovery delay and represents the upper bound energy cost for individual nodes. It is desirable to minimize this cost as well to reduce the recovery completion delay since sensor nodes need to buffer their data until the connectivity is restored. Therefore, the quicker the connectivity is restored, the more attractive the solution would be.
- *Coverage*: This metric measures total area sensed by the nodes in network. The higher the coverage the better the data accuracy and fidelity from the region.

Algorithm 6 federation $\{Par_i\}$

```
1: for  $j = \{1, 2, \dots, k, j \neq i\}$  do
2:    $e_{Par_j} = K_{Par_j}$ 
3:    $P(K_{Par_j}) = \frac{e_{Par_j}}{(b_i - b_{ij} + 1) * n}$ 
4: end for
5:  $e_{Par_i} = K_{Par_i}$ 
6:  $b'_i = |Par_i| - \#intersect\_nodes_{Par_i}$ 
7:  $P(K_{Par_i}) = \frac{e_{Par_i}}{(b'_i + 1) * n}$ 
8: if  $\exists P(K_{Par_j}) > P(K_{Par_i}), j = \{1, 2, \dots, k\}, i \neq j$  then
9:   while  $distance_{Par_j - Par_i} > r$  do
10:    relocate  $node_s \in Par_i$ , not  $\exists node_t \in Par_i, t \neq m, distance_{Par_j - Par_t} <$   

     $distance_{Par_j - Par_m}$ 
11:   end while
12: end if
13: if Sink Connectivity not Achieved then
14:   call computeEquilibrium $\{node_a\} \forall \in Par_j \cup Par_i$ 
15:   call federation $\{Par_j \cup Par_i\}$ 
16: end if
```

5.4.3 Baselines

In order to evaluate the performance, the proposed approach is compared with four other approaches. First, we compared it with the centralized sub-optimal solution presented in [16]. As the number of sensor to be deployed is limited with this approach, this comparison will be separate from others.

We also compared the proposed approach with two other centralized heuristics, DeCH and DiCH as described in Section 5.2.1 and 5.2.2 respectively. Note that sink node may also be relocated for recovery in these approaches. We also compared the proposed approach with the Route Based Distributed Heuristic (RDH) as described in [49]. And finally, we also compared the approach with the Relay Based Approach (RBA) presented in Chapter 4.

In the graphs given in 5.4.5, RDH is represented as *RDH*, RBA is represented as *RBA*, centralized heuristic mentioned in 5.2.1 is represented as *DeCH*, the other centralized heuristic mentioned in 5.2.2 is represented as *DiCH* and the proposed GTDH is represented as *GTDH*.

5.4.4 Performance Comparison Including the Sub-Optimal Solution

First, we provide performance comparison with the sub-optimal solution described in Chapter 3 which cannot support more than 30 nodes [16]. Hence, we used the same topologies as in [16]. The topologies with the highest number of nodes (e.g., 30) are selected. The number of partitions is picked as 3. Results on the travel distance and maximum travel distance are depicted in Table 5.2. Sub-optimal approach has the best performance as expected since it is centralized and requires complex computations. Another centralized approach, RBA also outperforms the proposed approach. It can also be observed that having been outperformed by the sub-optimal solution and RBA, *GTDH* not only minimizes the total movement cost of the network but also minimizes the maximum movement cost of an individual node when compared with *DeCH* and *DiCH*.

Table 5.2: Optimality of the Given Approaches

APPROACH	Total Travel Distance	Max Travel Distance
Sub-Optimal Solution	145.0819	45.9603
RBA	159.5592	72.9749
GTDH	170.812789	79.72928058
DeCH	225.0495653	101.8064736
DiCH	183.5669469	90.83384253

5.4.5 Comparison Excluding the Sub-Optimal Solution

In this subsection, we report the performance results by excluding the sub-optimal solution and comparing with other baselines by considering much larger networks that can support up to 200 nodes.

5.4.5.1 Total Travel Distance

We first conducted experiments for assessing the total travel distance assuming 2 partitions and varying the number of nodes from 50 to 200. Figure 5.3 demonstrates that the proposed *GTDH* performs almost the best among all the approaches. It outperforms *RDH* significantly. Our approach also outperforms the centralized *DeCH* by reducing the cost more than %60. Compared with *DiCH*, it performs almost identical. It can be

observed that while *GTDH* has a similar cost pattern with *DiCH* and *RBA*, *RDH* has a similar cost pattern with *DeCH*. Before starting to decrease, the recovery cost increases for *RDH* and *DeCH* until reaching a threshold value. This threshold value identifies the breakpoint where node degrees are high enough to compensate node failures. On the other hand, *RBA*, *GTDH* and *DeCH* perform well enough to ignore such a threshold.

Now, we elaborate on the behavior of the approaches. *RDH* performs worst since the nodes have very limited information about other nodes, and the recovery effort depends on the former paths between the partitions. Since these paths do not follow the minimum distance path between two locations, the path lengths are far behind being optimal. Furthermore, disallowing the movement of the sink node constraints the movement of the partition which the sink node belongs to. Thus, other partitions are enforced to move towards the sink node which may eventually be inefficient in terms of travel cost.

DeCH performs better than *RDH*. This is expected since *DeCH* is a centralized approach and can determine the path which incurs minimum cost when taken to connect partitions. However, it performs worse than both of the *GTDH* and *DiCH*. This is due to the difference in the sorting procedures followed. Depending on the node position, *DeCH* may consider a node before considering its neighbors and causes redundant movements.

Having been superior to *GTDH* and *DiCH* in Section 5.4.4, it can also be observed that *RBA* performs worse than *GTDH* and *DiCH*. This can be attributed to the better performance of *GTDH* and *DiCH* in denser topologies.

GTDH and *DiCH* performs almost identical. This is expected since only two partitions are considered in this experiment. Both approaches connect the partitions along the path with the minimum distance even though slight differences can be observed. This slight difference can be attributed to the fact that *DiCH* relocates nodes from the smaller partition towards the bigger partition based on the number of nodes they have. On the other hand, *GTDH* picks the partition with the smaller equilibrium. Thus, while cost of the cascaded movement is more in some cases for *GTDH*, it is observed to be less in some others. Node

degrees of the nodes which performs movement is a key factor on the cost since it is closely related with further movements needed to maintain connectivity of the other nodes.

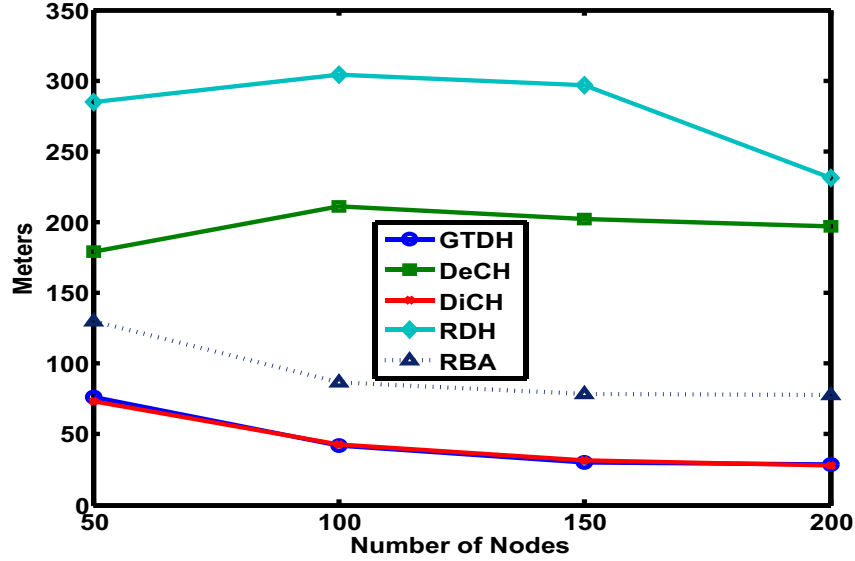


Figure 5.3: Total travel distance with varying number of nodes, # of partitions is 2.

To measure the performance of the mentioned approaches with the increasing number of partitions, the number of nodes is fixed to 150 and the number of partitions is varied from 2 to 5. Total travel distance performed by different approaches is depicted in Fig. 5.4.

The results in Fig. 5.4 shows that the increase in the number of partitions always increases the recovery cost no matter which approach is used. This is due to the increased effort to connect the partitions.

Again, *RDH* performs worst among all because of the same reason explained above. *DeCH* performs better than *RDH* but still performs worse than *GTDH* and *DiCH*. This is due to the ordering procedure of *DeCH* which considers partitions according to their sizes and the ordering of the nodes belong to these partitions.

In this case, *GTDH* performs slightly worse than *DiCH*. This is due to the distributed nature of this approach. Even though it performs identical to *DiCH* when only 2 partitions exist in the system, it may cause redundant movements when connecting partitions from more than two directions due to incomplete information about the partitions. *RBA* also closes the performance gap with *DiCH* and *GTDH* when the number of partitions is

increased.

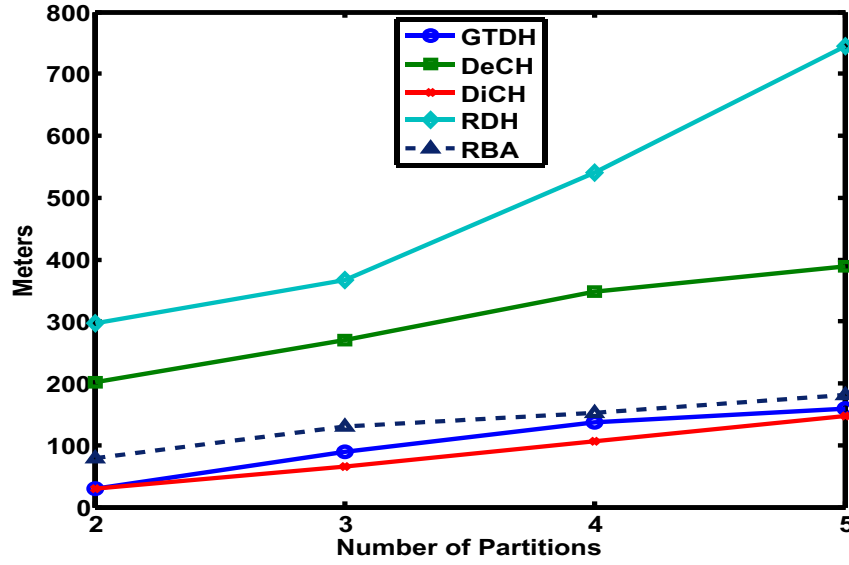


Figure 5.4: Travel travel distance with varying number of partitions, # of nodes is 150.

5.4.5.2 Maximum Travel Distance

For assessing the maximum travel distance of a node, we conducted experiments with varying number of nodes as seen in Figure 5.5. It can be observed that the maximum distance traveled by a node always declines when the number of nodes is increased in the network. This is expected since with the deployment of additional nodes, movement cost is shared among more nodes which eventually decreases the maximum movement cost of an individual node. Nonetheless, we would like to note that the decrease pattern in distributed approaches is different than the centralized approaches. Cost for distributed approaches decreases a little sharper which shows that they are more sensitive to the additional node deployment.

When looking at the performance comparison, we see a similar pattern as in the case of *Total Travel Distance* metric. *RDH* performs worst since the nodes have limited knowledge about the other nodes. *DeCH* performs better than *RDH* but worse than *GTDH* and *DiCH*. *RBA* has the worst performance initially and remains worse than *DeCH* for increasing number of nodes.

The only different behavior from Figure 5.3 is on the performance of *GTDH* and *DiCH*. While the maximum movement cost of *GTDH* is higher than *DiCH* for 50 nodes, the performance gap between the two approaches closes when the number of nodes is increased to 100 and more. This can be attributed to the fact that *GTDH* benefits from increased node density more compared to *DiCH* due to increased certainty about the partitions via boundary nodes.

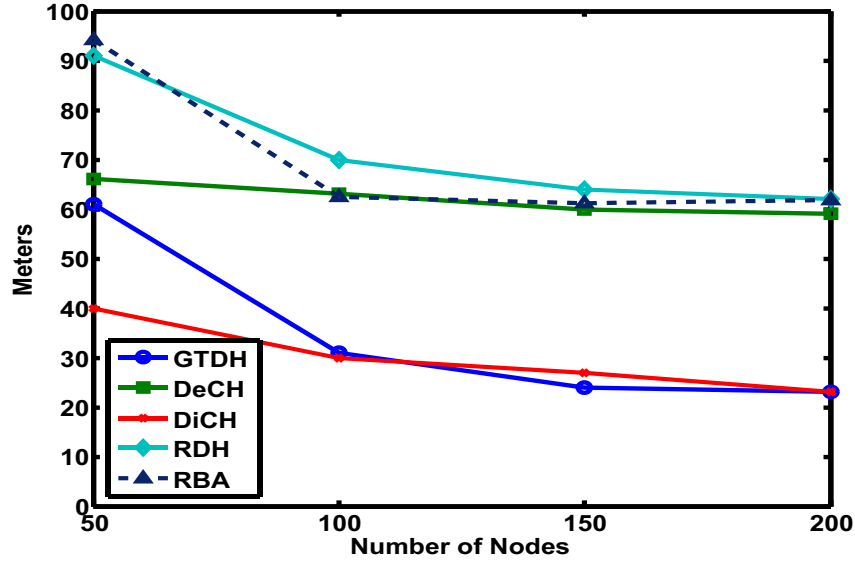


Figure 5.5: Maximum travel distance with varying number of nodes

We repeated the same experiments with a fixed number of nodes and varying number of partitions as shown in Figure 5.6. When the number of partitions is increased, the maximum distance between the partitions is expected to be higher which increases the maximum movement for all of the approaches.

Interestingly, the results show that *DeCH* does not perform as good as *RDH*. Considering the fact that *DeCH* outperforms *RDH* in terms of total travel distance for increasing number of partitions, we can speculate that *DeCH* cannot distribute the movement cost uniformly among the nodes when the number of partitions is high in the network.

Again, *RDH* performs slightly worse than *DiCH* due to its distributed behavior (i.e., some redundant movements are expected with the increased number of partitions).

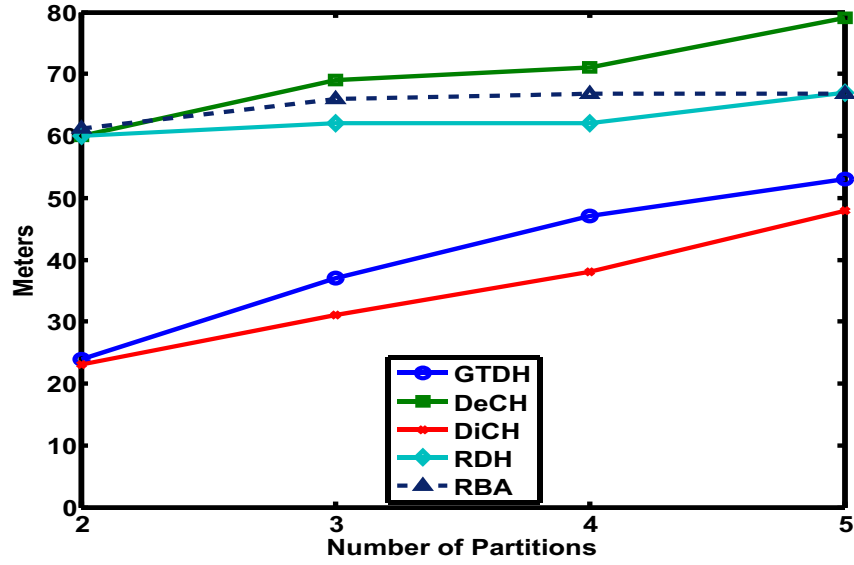


Figure 5.6: Maximum travel distance with varying number of partitions

5.4.5.3 Coverage

In order to determine how each approach improves the coverage after the partitioning, we conducted experiments with varying number of nodes and partitions as seen in Table 5.3 and Table 5.4 respectively. In these figures, we also demonstrate the coverage of the network before (i.e., Initial Coverage) and after (i.e., Coverage After Failure) the partitioning occurs. We observed that *GTDH* along with *RDH* and *DeCH* performs almost identical and bring the network coverage almost to its very initial stage. Despite its movement performance, *DiCH* provides the worst network coverage which is even worse with the increased number of partitions.

Table 5.3: Network coverage (m^2) with varying number of nodes when different approaches are applied.

# of Nodes	50	100	150	200
Initial Coverage	58.373	100.826	137.910	177.748
Coverage After Failure	24.585	43.028	57.086	69.507
GTDH	54.674	97.204	134.696	173.828
RDH	52.965	97.126	134.750	174.204
DeCH	54.718	97.680	135.415	174.890
DiCH	47.975	84.225	116.257	150.005

The performance of the mentioned approaches with the increasing number of partitions is depicted in Table 5.4. It can be observed that *RDH*, *GTDH* and *DeCH* perform similar and outperform *DiCH*. The network coverage declines for all of the approaches when number

of partitions increases. However, the performance decline of *DiCH* is more dramatic and the performance gap between the other approaches widens.

Table 5.4: Network coverage (m^2) with varying number of partitions when different approaches are applied.

# of Partitions	2	3	4	5
Initial Coverage	137.415	137.415	137.415	137.415
Coverage After Failure	57.086	48.954	37.509	31.802
GTDH	134.696	133.770	133.308	132.521
RDH	134.750	133.661	132.961	132.314
DeCH	135.415	134.301	133.998	133.338
DiCH	116.257	109.957	103.401	98.634

5.5 SUMMARY

In this chapter, we tackled the problem of connectivity restoration for a number of partitions in delay tolerant WSNs using mobility. We focused on a distributed solution since the solution should be self-organizing. Our approach was based on using Game Theory among the partitions of the WSN with a cooperative network game. Game Theory is used to improve the decision making process of the partitions when determining if the movement is necessary and the direction if so. Since the nodes can only access the nodes if they are in the same partition, a probabilistic model is applied to estimate the equilibrium of the other partitions based on the former paths and node degrees.

The experiments have revealed that the proposed Game Theoretic approach performs very close to the centralized sub-optimal solution and scales much better than the sub-optimal solution. The proposed approach is promising since the approach is distributed and self-healing in nature yet it performs almost the best in terms of total travel distance when compared to centralized approaches. In terms of recovery delay, its performance is still satisfactory since it is only a little worse than one of the centralized heuristics.

CHAPTER 6

RELAY PLACEMENT FOR RESTORING CONNECTIVITY IN PARTITIONED WIRELESS SENSOR NETWORKS UNDER LIMITED INFORMATION

In this chapter, we present two self-deployment strategies for relay node placement problem by assuming that the relays are to be populated to a safe place in the application area and then relocated to their final locations to connect partitions without the knowledge of the damage scale and the remaining nodes. The first approach is based on virtual force-based movements of relays polarized with the same magnetic pole. The second approach exploits Game Theory where the partition to be connected with is determined by the leader relay nodes based on the probability distribution function (pdf) of the partitions. The performance of both approaches is evaluated via extensive simulations. Game-theoretic approach has been shown to outperform force-based approach as well as a centralized approach under most of the conditions. This chapter is organized as follows. Section 6.1 states our assumptions and defines the problem. Section 6.2 describes the force-based approach. In Section 6.3 game-theoretic approach is explained in details. In Section 6.4, experimental evaluations and the results are presented. Finally, Section 6.5 concludes the chapter.

6.1 PRELIMINARIES

6.1.1 Assumptions

We assume a WSN comprised of sensors which are stationary and randomly deployed into the application area. Besides the sensors, a sink is also present to collect the data from the sensors. We assume that the sink is free from failures but the sensors may be damaged at an arbitrary time. Due to failures or initial deployment, we assume that the network is split to multiple disjoint partitions. The sensors can reach other sensors within the same

partition but not the rest of sensors in other partitions. Thus, sink node can only collect data from the sensors within its partition. We assume a recovery plan is initiated, once the partitioning is detected (e.g., by the sink node or collaboratively [49]).

As part of the recovery plan, it is assumed that mobile relay nodes are deployed to the application area in order to connect the partitions. We assume relays are resource rich in terms of battery power, memory and computation. Limited number of relays are deployed at once and multiple deployments are possible if the recovery cannot be completed with the given relays. We assume the relays are deployed to a safe region with a UAV. The application area is assumed to be obstacle free such that the relays can move on demand without encountering any obstacles. Initially, the relays may form a single or multiple partitions among themselves. Our approaches can work for both cases. In the latter case, the same approach runs on multiple partitions rather than a single partition. Note that even if there is a partition with a single node, with the additional node deployment in batches, this node will be connected to a partition eventually.

We also assume that the relays can determine their locations and identify the existence of other nodes in different partitions by exploiting their visual CMOS cameras/sensors [50][51] so that they can decide for the movement direction. [52] is an example robot equipped with such a camera. Identification of objects under the camera's field-of-view has been studied in image processing and recently been applied to resource constrained wireless visual sensor network [53]. The idea is to use frame differencing to extract the background. In some works, even the location of the identified objects can be determined with some computations [54].

A sample configuration for the network state after a failure is illustrated in Fig. 6.1.

6.1.2 Problem Definition

Formally, we define the problem as follows: A connected network with s nodes is given such that $\forall Node_i$ is accessible by $\forall Node_j$ for $1 \leq i \leq s$ and $1 \leq j \leq s$ either directly

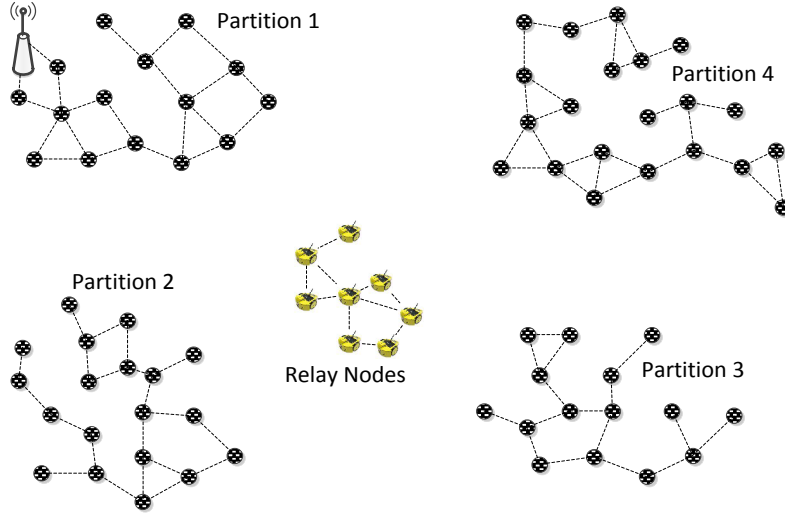


Figure 6.1: A partitioned WSN with a set of relay nodes.

or through other nodes. At an arbitrary time, $1 \leq f \leq s - 2$ nodes are removed from the network creating $k > 1$ partitions. Then n relay nodes are dropped to the damaged region. Our problem can be stated as: “Given a set of r sensors which is present in k partitions where $r = s - f$ and a set of n relay nodes, provide a distributed repositioning solution which integrates k partitions into a single connected network comprising r sensors and n relay nodes while minimizing n and $\sum_{i=1}^n d_i$ where d_i is the total travel distance of $Relay_i$ ”. Note that if n relay nodes do not guarantee connectivity, a new batch of n nodes can be added until the connectivity is restored.

6.2 FORCE BASED APPROACH

In this section, we provide an intuitive distributed approach to address the aforementioned problem.

6.2.1 Approach Description

The proposed approach is based on the idea of magnetic repelling forces in Physics which has been used in a number of coverage problems before [55][56]. Virtually all the relay nodes are assumed to be polarized with the same magnetic pole and push each other

as illustrated in Figure 6.2(a). To make the approach local, the pushing force is limited with the transmission range of the relay so that a force is applied to a node only from its neighbors. A node may have multiple neighbors pushing to different directions as seen in Figure 6.2(b). However, movement is not initiated until all the neighbors' forces are considered.

In this approach, each node collects the location information from its neighbors through local messages. Once the locations of the neighbors are collected, each node calculates the net pushing force applied by different neighbors. The closer the neighbor is, the higher is the repelling force. As represented in Figure 6.2(c) each neighbor's force is added by vector operations as in Physics and a new location is determined for the node to move. Note that if this location is out of transmission range of all of its neighbors, the new location is adjusted accordingly to stay within the transmission range of the closest neighbor to this new location.

Each movement changes the balance of the forces among the neighbors and triggers a new movement which is performed at a new iteration. At each iteration, the relay network becomes more stabilized and eventually the net forces will be small enough to be ignored and stop the process. When being spread, relay nodes expect to connect to partitions scattered around the network. Connectivity with a partition gets restored if a relay node moves close enough to a partition such that the distance between the relay and the partition (i.e., the closest sensor that belongs to that partition) is less than the transmission range. When such a partition is hit, the sensors in that partition which are within the transmission range of a relay node may also apply repelling forces back to the relay node. Recall that, repelling force is only considered for relay nodes and not applicable among the sensors. However, in order to stop the expansion towards a connected partition, sensor nodes in the periphery of a connected partition apply force to the relays. Similarly, for the nodes moving out of the damaged area (e.g., the ones which do not hit any partition), we apply repelling forces from the boundary of the application area. This will just require the boundaries and

shape of the monitored region to be known in advance.

If there are still partitions that remain disconnected after the relay network is stabilized (which can be determined by the sink), a new batch of relays will be deployed. Then the process will continue in the same manner until all the partitions are connected. The detailed pseudo-code for the algorithm is provided in Algorithm 7. This code is run by all relays in a distributed manner. Since only local information from neighbors within one hop is used, the approach is considered as distributed by its nature. The algorithm analysis in terms of message and movement complexity is provided in the next subsection.

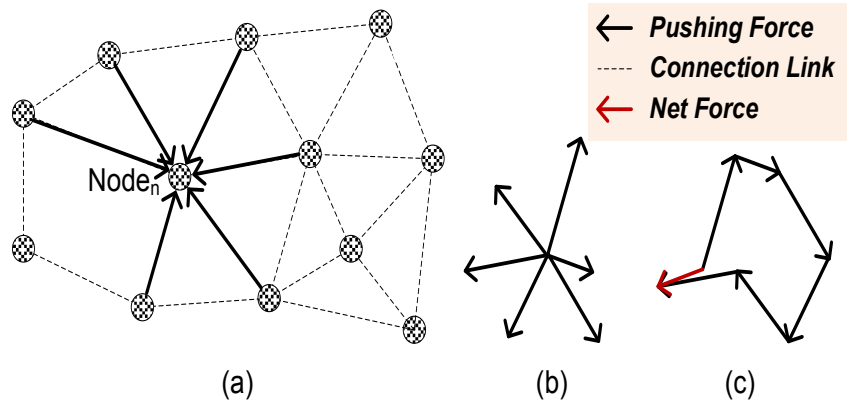


Figure 6.2: The pushing forces applied to $Node_n$ by its neighbors (a), calculation of forces on $Node_n$ (b) and calculation of net force on $Node_n$ (c)

6.2.2 Notation

Notations used in this section are listed in Table 6.1.

6.2.3 Algorithm Analysis

In this section, we analyze the algorithm in terms of message, computation and movement complexity in the worst-case scenarios. We consider the cases where connectivity is guaranteed with the used relays (i.e., in one batch).

Table 6.1: Notation for FBA

<i>Acronym</i>	<i>Explanation</i>
$node_i$	Sensor node i
Par_i	Partition i
k	# of partitions in the network
R	Set of relay nodes
n	# of relay nodes in the network
i	# of iterations before convergence
N	Set of neighbors
N_i	Neighbor i
m	Slope between the relay node and the given neighbor
d	Distance between the relay node and the given neighbor
f_{x_i}	Force applied by N_i in the x axis
f_{y_i}	Force applied by N_i in the y axis
f_x	Force applied by N in the x axis
f_y	Force applied by N in the y axis
l	Distance between given partitions

6.2.3.1 Message Complexity

Since the approach is distributed, message exchanges occur among the relay nodes. In this subsection, we analyze the worst case overhead due to message exchanges. This overhead translates to energy consumption of the relay nodes. We would like to mention that sensors within the partitions may also push the relays and thus send messages. However, the messaging cost can be eliminated if HELLO packets from the sensors are used for the detection of their presence and hence calculation of pushing force. Therefore, we only include message complexity analysis for the relays.

Theorem 1. Assuming a network with a total of $n = |R|$ relays deployed, the worst case total message complexity of the forced based approach is $O(n * i)$ where i represents the number of iterations before convergence.

Proof. The forced-based approach is fairly simple with a single type of message: Position notification (PN). Most of the routing algorithms provide a mechanism to obtain the neighbors list within one hop.

Therefore, each relay broadcasts its location and ID at each iteration. Using the

Algorithm 7 FBA()

```
1: partitions = Par1, Par2, ..., Park
2: while |partitions| > 1 do
3:   for i = {1, 2, ..., |N|} do
4:     sendMessage(Ni, id, location)
5:   end for
6:   for i = {1, 2, ..., |N|} do
7:     (id, location)Ni = recvMessage()
8:   end for
9:   //Wait until receive all or timeout
10:  for i = {1, 2, ..., |N|} do
11:    fx = fy = 0
12:    neighbor = Ni
13:    m = slope(location, neighbor)
14:    d = distance(location, neighbor)
15:    fxi = computeForce(locationx, neighborx, TR - d, m)
16:    fyi = computeForce(locationy, neighbory, TR - d, m)
17:    fx = fx + fxi
18:    fy = fy + fyi
19:  end for
20:  move(locationx + fx, locationy + fy)
21: end while
```

location and ID information of the neighbors, the net force applied to a relay is computed. Movement is postponed until receiving a PN message from all of the neighbors. Note that to avoid delays which may be caused due to message loss and neighbor list conflicts, a timeout is defined to allow movements without reception of the messages from all of the neighbors. Net force is computed upon reception of the PN messages and movement is performed. Depending on the message arrival times, neighboring relays can initiate movement simultaneously. The worst case total PN message complexity is $O(n * i)$ where i represents the number of iterations before convergence. $\square$

6.2.3.2 Computational Complexity

Considering limited computational power of the relays, it is important to minimize the complexity of the approaches. Otherwise, computations may not be completed in a reasonable amount of time and the recovery may be delayed. If the complexity exceeds the

computational power, computations may never be completed due to run-time errors and recovery will not be achieved. In this subsection, we analyze the worst case computational complexity of the forced-base approach.

Theorem 2. Assuming a network with a total of $n = |R|$ relays deployed, the worst case computational complexity of the forced-based approach is $O(n)$.

Proof. In a distributed implementation, all relays are responsible from computing the net force applied to them. In the worst case, a relay node will have $n - 1$ immediate neighbors within one hop. Considering the fact that a single pass in the neighbors list is enough to compute the net force, computational complexity of the forced based approach can be expressed as: $O(n)$. $\square$

6.2.3.3 Movement Complexity

In this section, we analyze the worst case travel distances for the approach. We look at both the total travel of all the nodes and the maximum travel among all the nodes.

Theorem 3. The worst case maximum travel distance in force-based approach for all the nodes is $l - d$.

Proof. The worst case occurs when there are two partitions located at a distance of l from each other as seen in Fig. 6.3. In this case, assuming all the relays are placed next to one of the partitions on a line and they push each other on this line, then a relay will have to travel up to the next partition. The maximum movement distance in this case will be as long as the length of $l - d$ where d is the transmission range. $\square$

Theorem 4. The worst case total travel distance for force-based approach is $\sum_{i=1}^{\lceil \frac{l}{d} - 1 \rceil} (l - id)$.

Proof. The case for the total movement distance is tightly correlated with the maximum movement distance and given in Fig. 6.3. With the deployment of additional relays, relays keep expanding until both of the partitions are connected. Thus, all the relays will be in

motion unless the connectivity with the furthest partition is restored. Therefore, the worst case for the total movement distance occurs in a scenario given in Fig. 6.3 where all the nodes are deployed on a line at the same location next to the one of the partition. Relay will start pushing themselves until one of the relay reach to the other partition (i.e., come close with a distance of d). In this case, one of the relays will move a distance of $l - d$, the next one will move $l - 2d$ and so on. If we add all the movement distances, then we get $\sum_{i=1}^{\lceil \frac{l}{d} - 1 \rceil} (l - id)$ independent from the relay count. $\square$

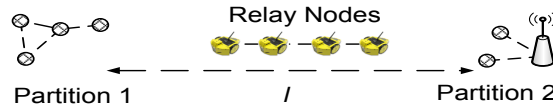


Figure 6.3: Scenario causing the worst case movement for FBA and GTA.

6.2.3.4 Convergence of the Algorithm

Several factors affect the number of relays deployed to the application area including the number of partitions, number of nodes within the partitions, location of the nodes in each partition, positions of the relays deployed to the area, etc. We define a set of rules to guarantee convergence of relay movements and provide network wide recovery. First, we constraint the movement of relays within the application area so that relays do not cross the application area border and leave the area. Once the relay determines the target position for movement, it checks whether the destination remains within the application area or not. If the destination is out of the application area, relay updates the movement target accordingly and stops at the border of the application area.

Another constraint on the relay movement is prohibiting movement of the relay unless the net force is more than the movement threshold. Since each of the relays applies a force to its neighbors, movement will not stop until all the relays reach equilibrium. While reaching equilibrium, movement distances will decrease in each iteration until relays stop permanently. To accelerate reaching the equilibrium state, we define a threshold to limit movements so that we do not allow movement of a relay unless the net force is above the

threshold. This threshold will guarantee convergence of the algorithm.

This force-based approach is distributed in the sense that the relay nodes only know their neighbors and talk to those neighbors to compute the net repelling forces. However, this approach requires every relay to move based on the forces and thus may not be efficient in terms of movement costs. Therefore, we propose a more movement efficient approach based on Game Theory in the next section.

6.3 GAME THEORETIC APPROACH

6.3.1 Model

We use the same model explained in Chapter 5. Please refer to Chapter 5 for details.

6.3.2 Algorithm and Pseudo Code

In this subsection, we present a distributed algorithm to restore connectivity with the partitions by relocating the post-deployed relay nodes towards the partitions. Given that the approach is distributed and the deployed relay nodes have no previous information about the network structure and damage, a probabilistic approach is pursued to determine the direction of the movement.

The relays collaboratively estimate the equilibria of the partitions in the system and define their pdfs based on the estimated equilibrium. Note that the relays do not have any information about the location of the partitions when deployed and to estimate the locations of the partitions, primitive cameras on the relays are exploited. Each relay can roughly identify the nodes in the partitions via detecting the sensor nodes that appear in its field-of-view (FoV) (i.e., the pie-shaped area a camera can cover) along with the estimate of their locations [54]. Each relay can send the collected sensor count, their locations and its own location to a representative within its network (i.e., the relays that are connected and form a network initially). The representative can eliminate the overlapping FoVs and exclude the repetitions in the node count and estimate a sensor count for a particular

partition of sensors.

The number of sensor nodes identified for each partition is used when computing the equilibrium of that partition. Due to the possibility of blocking from existing relays and sensors, full view may not be possible and the number of nodes identified may not be accurate but still it will give an idea of the degree. Note that the proposed approach can run with the partial information to define a pdf for each partition.

Once the pdfs are defined, relays move one by one towards the partition with the highest pdf until connectivity is restored. The movement locations are again estimated using cameras. When the connectivity is restored with a partition, relays determine the next partition to connect based on the pdf if such a partition exists. The relays keep relocating until all the partitions are connected. Note that additional relays are deployed if the operation is suspended due to the insufficient relays in the system.

Equilibrium function e_{Par_i} of partition Par_i can be stated as below for $|AN|$ denoting the total number of nodes in the relay network:

$$e_{Par_i} = \sum_{w=1}^{|AN|} FoV(\exists node \in Par_i) \quad (6.1)$$

Pdfs of the partitions are defined once the equilibria of the partitions are computed. Pdf $P(K_{Par_i})$ of partition Par_i can be stated as below for k denoting the total number of partitions in the system:

$$P(K_{Par_i}) = \frac{e_{Par_i}}{\sum_{w=1}^k e_{Par_w}} \quad (6.2)$$

The pseudo-code for the connectivity recovery process is given in Algorithm 8 which is named as GTA. The approach is regarded as distributed in the sense that complete

information regarding other partitions is not required and movement decisions can be made with incomplete information.

Algorithm 8 GTA()

```

1: partitions = Par1, Par2, ..., Park
2: while |partitions| > 1 do
3:   broadcastMessage(LS, id, TTL = |AN| - 1)
4:   message = recvMessage(LS, id', TTL')
5:   leader = (id' < min)? id' : min
6:   forwardMessage(LS, id', TTL' - 1)
7:   for i = {1, 2, ..., k} do
8:      $e_{Par_i} = \sum_{w=1}^{|AN|} FoV(\exists node \in Par_i)$ 
9:   end for
10:  sendMessage(PIC, leader, ePar, location)
11:  //Only leader runs this part
12:  for i = {1, 2, ..., |AN - 1|} do
13:    recvMessage(PIC, ePar, location)
14:  end for
15:  for i = {1, 2, ..., k} do
16:     $P(K_{Par_i}) = \frac{e_{Par_i}}{\sum_{w=1}^k e_{Par_w}}$ 
17:    targetP = ( $P(K_{Par_i}) > maxP$ )? Pari : maxP
18:  end for
19:  for i = {1, 2, ..., |AN|} do
20:    closestNode = (distance(nodei, targetP) < minDist)? nodei : minDist
21:  end for
22:  //If closestNode is not the leader
23:  sendMessage(MN, closestNode, targetP)
24:  //Only closest node runs this part
25:  broadcastMessage(MN, id, location, targetP)
26:  move(targetP)
27:  while distance(targetP, AN) > TR do
28:    //Update closest node
29:    broadcastMessage(MN, id, location, targetP)
30:    move(targetP)
31:  end while
32: end while

```

Below, we elaborate on the details of the message exchanges among the nodes within the GTA algorithm:

- *Leader selection (LS)*: A leader node is employed among the relays to compute the equilibria and pdfs of the partitions. Each relay is assigned with a unique ID and the

one which has the lowest ID is selected as the leader. In order to determine the node with the lowest id, each relay broadcasts an LS message with its id. For n relays in the relay network deployed in the last batch, LS message is sent with a TTL of $n - 1$. To minimize message complexity, LS message is dropped upon reception unless the ID of the message originating relay is lower. In the worst case, for n relays, sending $n - 1$ messages each, the worst case message complexity is $O(n^2)$. Note that other leader election algorithms are possible but the complexity will not be less than $O(n^2)$ [57].

- *Partition information collection (PIC)*: Equilibrium of the partitions is derived from the number of visible nodes in each partition. Existing relays and sensors may block the visibility of some of the sensors in the partitions and the number of nodes may not be determined accurately. GTA can run with the partial information though and sharing the information about the partitions will improve estimating the degrees of the partitions. Thus, each relay sends the number of visible nodes in each partition as well as the location information to the leader node. For n relays in the relay network deployed in the last batch, $n - 1$ relays will send a message to the leader. If all of the relays are within transmission range of the leader (i.e., best case), only $n - 1$ messages will be sent and message complexity will be $O(n)$. However, in the worst case, a line topology is formed by the relays and the leader node resides at the edge. In the worst case, $(n - 1) \times n/2$ messages are sent and the message complexity can be represented as $O(n^2)$.
- *Movement notification (MN)*: Once the pdfs for the partitions is computed by the leader node, the partition with the highest pdf is selected as the target partition to connect with. Since the leader already has the location information of other relays, it can determine the closest relay to the target partition. The relay with the minimum proximity to the target partition performs the first movement. Other relays, one by

one, follow the first relay as well until the recovery is completed. If the closest relay is not the leader node, a notification message is sent to the closest relay by the leader node. This message travels $n - 1$ hops incurring a cost with a complexity of $O(n)$ in the worst case. Before initiating the movement, closest relay sends MN message to its neighbors to notify its movement so that one of the neighbors can follow it until the recovery is completed. In the worst case, $n - 1$ neighbors can exist requiring $n - 1$ message to be sent. Thus, message complexity of this part can be expressed as $O(n + n)$ or simply as $O(n)$.

6.3.3 Algorithm Analysis

This section analyzes the game-theoretic approach using the same assumptions in Section 6.2.3.

6.3.3.1 Message Complexity

Game-theoretic approach requires message exchange to perform its operations. In this subsection, worst case overhead incurred due to message exchanges is analyzed. This overhead translates to energy consumption of the nodes.

Theorem 5. The worst case total message complexity of the game-theory based approach is $O(n^2)$ where n represents the initial number of relays.

Proof. In order to fulfill the desired functionality, GTA requires exchanging three types of messages: Leader selection, partition information collection and movement notification as explained before. The worst case message cost for these messages were computed as $O(n^2)$, $O(n^2)$ and $O(n)$ respectively. Thus, total message complexity for GTA is $O(2 * (n^2) + n)$ or simply as $O(n^2)$. $\square$

6.3.3.2 Computational Complexity

Limited computational power of the relays may delay or even impede the recovery. Thus, approaches must be developed considering the limitations on the computational power of relays. In this subsection, we analyze the worst case computational complexity of the game-theoretic approach.

Theorem 6. Computational complexity of the game-theoretic approach is $O(hw)$ where h and w are the height and width in terms of pixels for a frame taken by the camera of a sensor.

Proof. In the algorithm there are two types of computation. First, images are taken and some image processing is applied to extract the sensors, their count and locations. This processing purely depends on the number of pixels on the frame. Basically, all the pixels should be visited to gather the above information [54]. This brings a computational complexity of $O(hw)$ for each relay. In addition to this cost, there is a processing cost which can be computed at the leader node to determine the pdfs of the partitions. For k partitions, pdf of the partitions can be computed in $O(k)$. The partition with the highest estimated pdf can be determined with a single pass which has a complexity of $O(k)$ as well. Therefore, for the leader node we will have a total cost of $O(hw + k)$ which can converge to $O(hw)$ since k will be much less compared to h and w . $\square$

6.3.3.3 Movement Complexity

Excessive traveling distances impair the lifetime of the network. In this subsection, we analyze the worst case movement of game-theoretic approach. Both the total movement and the maximum movement of the nodes are discussed.

Theorem 7. In the worst case, maximum travel distance for GTA is $l - d$ where l is the distance between two partitions and d is the transmission range.

Proof. The worst case maximum travel distance for GTA is illustrated in Fig. 6.3. This case occurs when two partitions are located at a distance of l to each other in a line topology and the relays are deployed next to one of the partitions. The maximum travel distance in this case will be as long as $l - d$ so that the relay can get connected to the other partition. $\square$

Theorem 8. The worst case total travel distance for GTA is $\sum_{i=1}^{\lceil \frac{l}{d} - 1 \rceil} (l - id)$.

Proof. Therefore, the worst case for the total travel distance occurs in a scenario given in Fig. 6.3 where all the nodes are deployed on a line at the same location next to the one of the partition. Relay will start moving towards the other partition (i.e., come close with a distance of d). In this case, one of the relays will move a distance of $l - d$, the next one will move $l - 2d$ and so on. If we add all the movement distances, then we get $\sum_{i=1}^{\lceil \frac{l}{d} - 1 \rceil} (l - id)$ independent from the relay count. $\square$

6.3.3.4 Convergence of the Network

Unlike, *FBA*, in GTA relays move towards a partition directly by disconnecting with the rest of the relay nodes. The partition to be connected with is identified based on the pdf computations. As long as partitions exist, there will be a partition which dominates the partition of the relay nodes. Therefore, connection process will be pursued with the next dominating partition until no partition is left and overall network connectivity is restored. Every partition will be connected with the rest of the network as long as more relays are added whenever needed. The relays can detect every other partition that are within the FoV of their cameras.

However, game-theoretic approach may fail to ensure complete recovery in concave shaped deployments. The problem occurs when detecting the nodes of a partition masked by another partition. Such a scenario is given in Fig. 6.4. In the figure, Partition 1, may not be detected as a separate partition. While the probability of discovering such partitions can be improved by repeating the discovery process upon reaching the destination partition,

similar issues persist and thus detection of every partition may not be guaranteed. We leave these cases as a future work for investigation and do not tackle them in this dissertation.

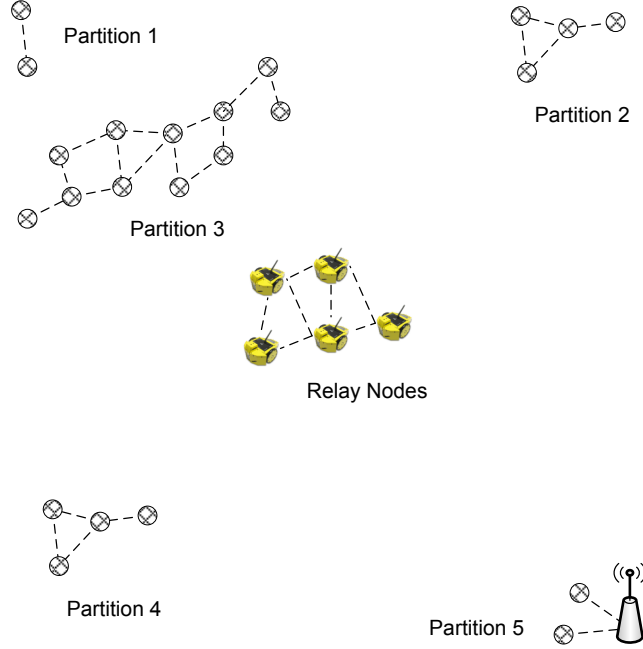


Figure 6.4: GTA may fail to ensure complete recovery for concave deployment.

6.4 EXPERIMENTAL EVALUATION

In this section, we first describe the experiment setup, performance metrics and the baselines for comparing with our approach. For each performance metric, the results are demonstrated and discussed. We also present experiments by considering the initial deployment location for the relays (e.g., center vs edge).

6.4.1 Simulation Setup

The performance of the presented approaches have been evaluated in a simulated environment. We consider the terrain as an obstacle free environment with a size of 800X800 meters. Transmission range is set to 50 meters for both the nodes and the relays. Two test cases are considered; node density and disaster scope. To test the impact of the node density, 50 to 200 sensor nodes are deployed in such a way that only 2 disjoint partitions are

formed in the WSN. To test the impact of the disaster scope, 150 sensor nodes are deployed to form 2 to 5 partitions. We also define two deployment patterns: fixed and decreasing relay count in each batch. In the fixed-size deployment, 5 new relays are deployed to the application area at each batch. In the decreased-size deployment, the number of relays deployed at each batch is decreased 1 each time starting from 5. Finally, 1 relay is deployed in each batch until the connectivity is achieved. The relay nodes are deployed at the center of the damaged region by default. Later, we also considered cases where the relays are deployed at the edges of the damaged region. 30 different topologies are created for each test case and for significance, the average is reported.

The following metrics are used to measure the performance of the presented approaches:

- *Total Travel Distance*: Total distance traveled by all the relay nodes. Movement demands high energy consumption and impacts the overall network lifetime. Therefore, it is important to reduce the travel distance.
- *Maximum Travel Distance*: The maximum distance traveled by an individual relay node. Relay nodes have limited energy supply which can be depleted upon excessive movement. Maximum travel distance impacts the lifetime of an individual relay node and hints about the delay associated with the recovery.
- *Total Relays Deployed*: Relay nodes are deployed in multiple batches if required. The number of relay nodes required to assure complete recovery must be minimized in order to reduce equipment and deployment costs.

In addition to force-based approach (represented as *FBA* in the graphs) and game-theoretic approach (represented as *GTA* in the graphs), we also implemented a centralized approach to be used as a baseline (represented as *CA*). The idea is to restore the connectivity by assuming that optimal relay count and locations are known in advance and the relays will

move to those locations from their initial locations where they are deployed. To determine the minimum relay count and the relay locations, we utilized one of the relay placement heuristics in the literature [58]. Once the optimal relay locations are known as a result of running this heuristic called CIST, then relays are deployed randomly (as done for *FBA* and *GTA*) to the deployment area. Deployment location of the relays is regarded as the initial location before movement and the locations obtained from CIST is regarded as the final locations after movement. Initial and final locations are paired based on the distance so that each relay moves to the closest unoccupied final position. This centralized approach minimizes the total movement and thus will be used as a benchmark to assess the performance of *FBA* and *GTA* in terms of movement distance. Note that *FBA* and *GTA* have two versions based on the deployment plans. As explained in Section 6.4.2.3, the relays are deployed in batches and the number of relays to be deployed at a later batch is either fixed or decreasing. Based on this strategy two versions of *GTA* and *FBA* are implemented. These are shown as *GTA-F*, *GTA-D*, *FBA-F*, *FBA-D* in the graphs where *F* represents fixed number of relays in deployment and *D* represents decreasing number of relays.

6.4.2 Experiment Results

6.4.2.1 Total Travel Distance

We first conducted experiments to measure the total travel distance and provided the results for the two test cases on total travel distance: node density and disaster scope in the following graphs respectively: Figure 6.5 and Figure 6.7. It can be noticed from Figure 6.5 that the node density has a positive effect on decreasing the overall movement cost of the relay nodes. Movement cost declines for all approaches. This decline is expected considering the possibly shorter distances among the partitions in denser networks. The minimum cost is achieved when the density of the network is maximized.

It can be concluded that *GTA* significantly outperforms *FBA* especially in sparse

networks. The performance gap declines in denser networks since the nodes in *FBA* move smaller distances as their count is increased within a fixed area. Excessive cost of *FBA* arises from the involvement of all the relay nodes in movement after each deployment until reaching a balanced state. This is not the case in *GTA* since relay nodes move individually as needed. Despite the little gap, both *GTA-F* and *GTA-D* also outperform *CA*. This is interesting given the fact that *CA* requires the least number of relays. Nevertheless, as will be detailed under Section 6.4.2.4, *CA* deploys less relays which causes the individual relays to move longer distances. In *GTA*, since there are more relays, shorter distances are traveled which eventually makes the total travel distance slightly less than that of *CA*.

When we compare the deployment patterns, it can be noticed that both of *FBA-F* and *GTA-F* perform better. While the gain is negligible for *GTA-F* over *GTA-D*, *FBA-F* performs significantly better than *FBA-D* especially in sparse networks. This can be attributed to the number of relays used in both deployment patterns. It can be observed that decreased size deployment requires less relays compared to fixed size deployment for both approaches. When more relays are present in the region, finding a closer partition will be easier due to spatial distribution of relays. As a result, the relays move smaller distances towards those partitions, reducing the overall movement. Since all the relays participate in the movement, the effect is more significant with *FBA*.

Next, we looked at the impact of the disaster scope on the total travel cost as illustrated in Figure 6.7. It can be observed that the travel cost increases for all of the approaches when the disaster scope expands as expected. This is due to the increasing number of partitions which need to be connected with the rest of the network. It causes more relay nodes to be deployed to complete the recovery and the total travel distance will be higher. As in the case of increased node density, we observe the same behaviors, e.g., *GTA* outperforms *FBA* and *CA* again in all conditions. The justification of these results is same as in the case of node density experiments.

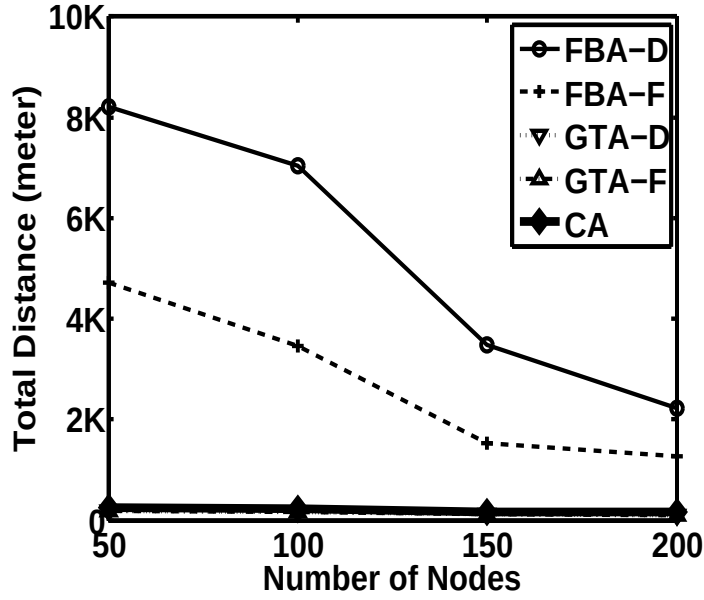


Figure 6.5: Total travel distance with varying number of nodes. The number of partitions is fixed to 2.

6.4.2.2 Maximum Travel Distance

We assessed the burden of the proposed approaches on the individual relay nodes as well. It should be noted that if the overall movement cost for a single relay node is too high, relay node may fail to complete the movement due to battery depletion or may be subject to pre-mature failure after a short time once the recovery is completed causing necessity of another recovery. We discuss how the node density affects the maximum travel distance first. Figure 6.9 indicates that node density alleviates the maximum movement distance of the relay nodes. All of the approaches enjoy the decline of the cost. This can be attributed to the fact that the distance between the partitions is expected to be shorter when the density of the network is increased. This is analogous to the total travel distance cost discussed in the previous subsection.

We can notice that *GTA* performs considerably better than *FBA* particularly in networks with fewer nodes. When the number of nodes is increased, *FBA* takes advantage of the node density more compared to the *GTA*. Nonetheless, *FBA* has better performance in sparse and dense networks compared to *GTA*. As in total travel distance, *FBA* suffers

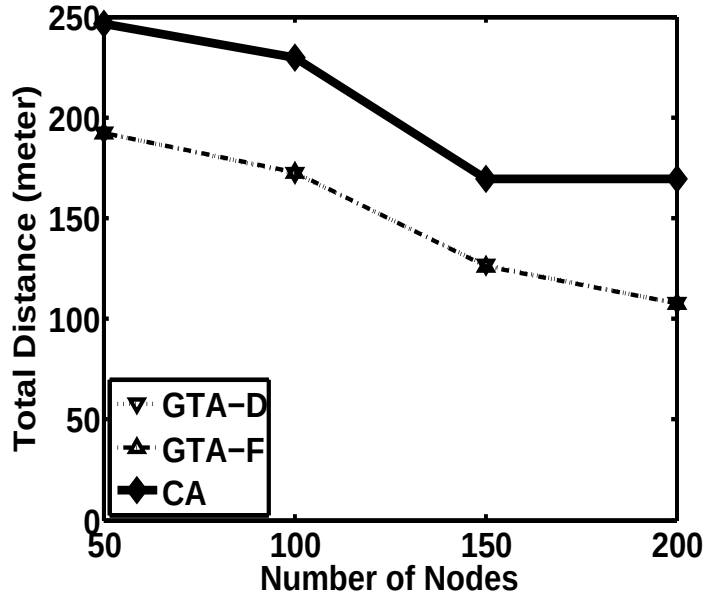


Figure 6.6: Total travel distance with varying number of nodes. The number of partitions is fixed to 2. Results in Figure 6.5 is magnified for *GTA* and *CA*.

from the involvement of all the relay nodes in movement causing the maximum travel distance to be higher. The slightly poor performance of the *CA* compared to *GTA* can be attributed to the number of relays used, deployment location of the relays and the nature of the approaches. In particular, when the number of relays is less, there is high chance that some of the nodes may travel longer distances.

When we compare the deployment patterns, it can be concluded that fixed size deployment is preferable to decreased size deployment in order to decrease maximum movement cost for both of the approaches. Again same pattern is observed as in the case of total travel distance metric and thus we do not repeat the same analysis again.

The impact of the disaster scope on maximum movement cost is illustrated in Figure 6.11. It can be recognized that the maximum movement cost increases for all of the approaches when the disaster scope expands. Having more partitions in the system increases the possibility of having a partition that needs to be connected with longer distance movements. Thus, maximum travel distance is higher when the disaster scope expands. *GTA* outperforms *FBA* and *CA* again in all conditions and fixed size deployment creates

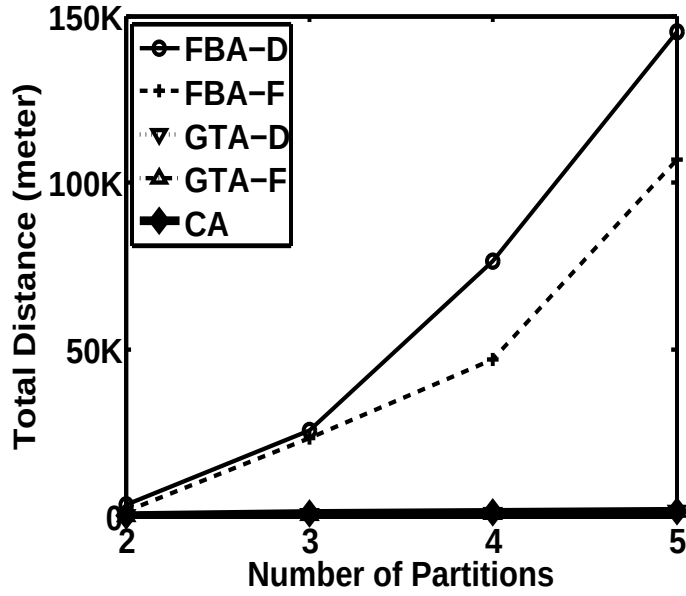


Figure 6.7: Total travel distance with varying number of partitions. The number of nodes is fixed to 150.

less movement cost compared to decreased size deployment as observed before in assessing the total travel distance.

6.4.2.3 Total Relays Deployed

For assessing the total number of relays deployed at the end of the batches, we conducted experiments considering the two test cases: node density and disaster scope. We discuss how the node density affects the deployment cost first. Figure 6.13 indicates that the relay count is decreasing for all of the approaches when the number of sensors is increased in the network. This is inevitable since in denser networks, the distance between the partitions is expected to be shorter and eventually less relays are required to complete the recovery.

When comparing the approaches, we can observe that *CA* outperforms the other approaches. This is expected, since *CA* is a centralized approach which aims to minimize the relay count. Nonetheless, *GTA* performs very close to *CA* and outperforms *FBA* particularly in sparse networks. This is because, the number of relays required to complete the

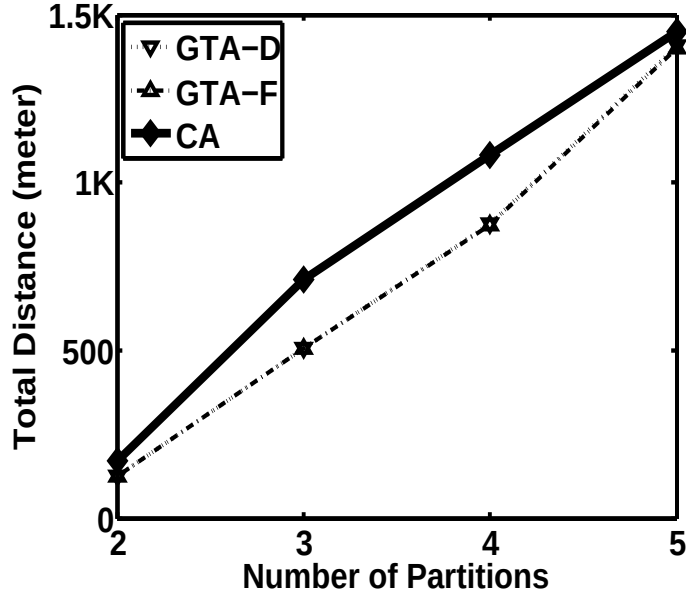


Figure 6.8: Total travel distance with varying number of partitions. The number of nodes is fixed to 150. Results in Figure 6.7 is magnified for GTA and CA.

recovery increases for *FBA* especially in sparse networks where the partitions are expected to be farther away from each other. In *GTA*, on the other hand, the relays move towards a particular partition at a time and thus the goal is achieved much quickly with less relays. Overall, when the performance of *GTA* in terms of total and maximum travel distance is also considered, we can see that *GTA* is best suited for recovery with its distributed nature and less movement overhead.

When the deployment patterns are compared, it can be observed that decreased-size deployment completes the recovery by using less relays compared to fixed-size deployment for both *GTA* and *FBA*. While the gap between the deployment patterns is minimal for *GTA*, it is considerably more significant for *FBA* due to the increased relay count. Better performance of the decreased-size deployment can be attributed to the fact that some of the relays deployed in the last batch may be redundant when fixed-size deployment is used. This is not the case in decreased-size deployment where final batches require only a single relay.

The impact of the disaster scope on the total relays deployed is illustrated in Fig-

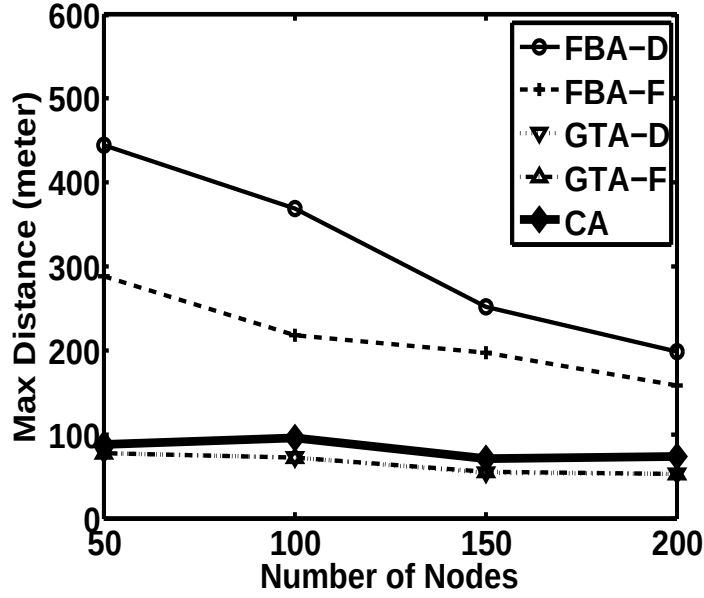


Figure 6.9: Maximum travel distance with varying number of nodes. The number of partitions is fixed to 2.

Figure 6.14. It can be observed that the number of relays to be deployed increases for both *GTA* and *FBA* when the disaster scope expands. This is expected since having more partitions in the network increases demand for more relays to complete the recovery due to increased need of connections between partitions. As in the case of the first experiment with varying node count, *GTA* outperforms *FBA* in all conditions and decreased-size deployment requires less relays to be deployed compared to fixed-size deployment. As expected, *CA* has superior performance compared to *GTA* and *FBA*.

6.4.2.4 Deployment Location

The initial deployment location for the relay nodes may affect the performance of the given approaches. To assess the impact of the initial location, we considered two different scenarios for relay deployment. In the first scenario, we assumed that the geographic center of the partitions can be estimated and the relays are initially placed or dropped at or around the center location. Note that this was the default scenario used in the previous experiments. The other scenario assumes that the relays are placed/dropped at

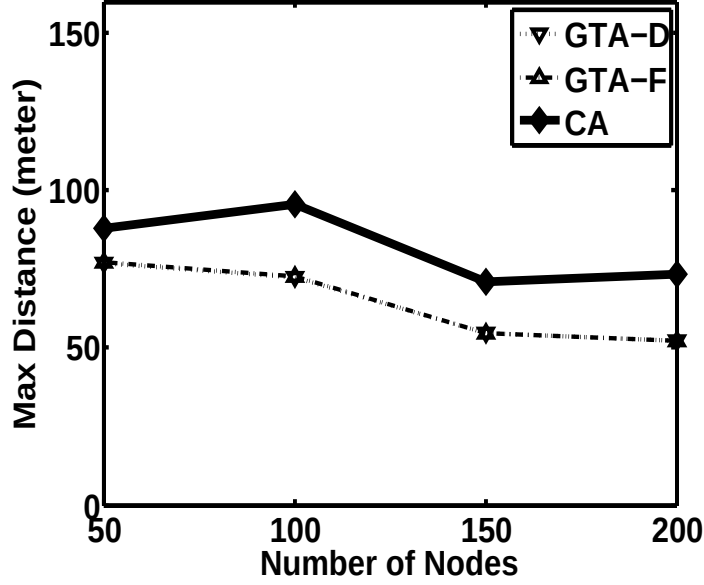


Figure 6.10: Maximum travel distance with varying number of nodes. The number of partitions is fixed to 2. Results in Figure 6.9 is magnified for GTA and CA.

one of the edges of the application area (e.g., the application area is not accessible due to hazards or threats). Given the poor performance of *FBA*, we only compared *GTA* and *CA* for the deployment location. The approaches using these scenarios are represented as $|ApproachName| - Center$ and $|ApproachName| - Edge$.

First, we looked at the total distance traveled by all the relay nodes by varying the node density and partition count. It can be noticed from Figure 6.15 that the deployment scenario has a significant influence on the selected approach. When the relays are deployed in the center, *GTA* performs better than *CA* even though the performance gap is not very high. On the other hand, when the relays are deployed at the edge, *CA* performs significantly better than *GTA*. This difference can be attributed to the nature of the approaches. Even though the first relays connecting different partitions may move longer in *GTA*, each relay's movement reduces until partitions are connected. On the other hand, *CA* usually connects different partitions one by one and the relays connecting different partitions are usually disconnected in their final positions and are far from each other. Thus, each relay may need to move as much as other relays and the movements of the initial relays do not

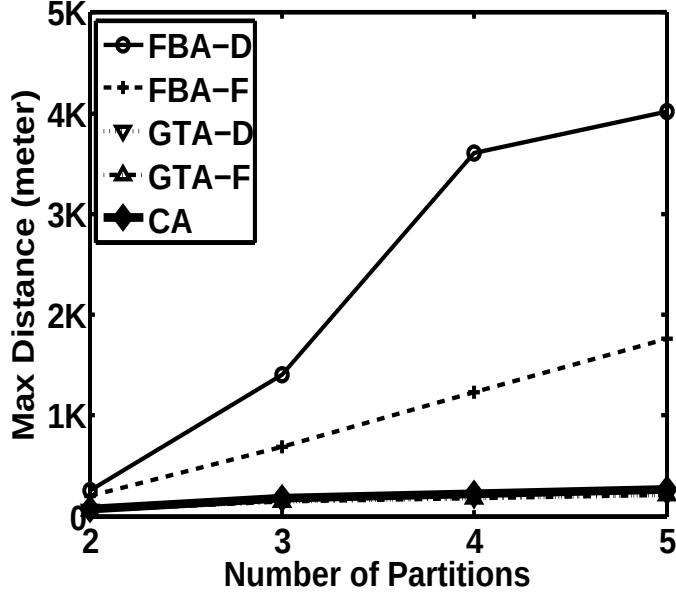


Figure 6.11: Maximum travel distance with varying number of partitions. The number of nodes is fixed to 150.

decrease the movement cost of the relays moving at later stages. *CA* has even better performance if the relays are deployed at the edge. In this case, *CA* exploits the advantage of the optimal relay positions to connect the partitions.

We also provide the impact of the disaster scope on the approaches in Figure 6.16. Deployment to the center decreases the cost for both approaches comparing with the deployment to the edge. This is due to closer distance to the final locations from the center of the application area. As in previous example, *GTA* outperforms *CA* for center deployment and *CA* outperforms *GTA* for the deployment at the edge.

The second metric we considered is the number of relays to be deployed. As mentioned earlier, *CA* determines the number of relays required to guarantee connectivity. We used this number as the initial number of relays to be deployed for *GTA* and added additional relays if required. Finally, we compared the number of relays required by these approaches. The experiment results indicated that no matter what deployment scenario is used, the number of relays required by *CA* is same as seen in Figure 6.17. On the other hand, the number of relays to be deployed declines significantly for *GTA* when the relays are deployed

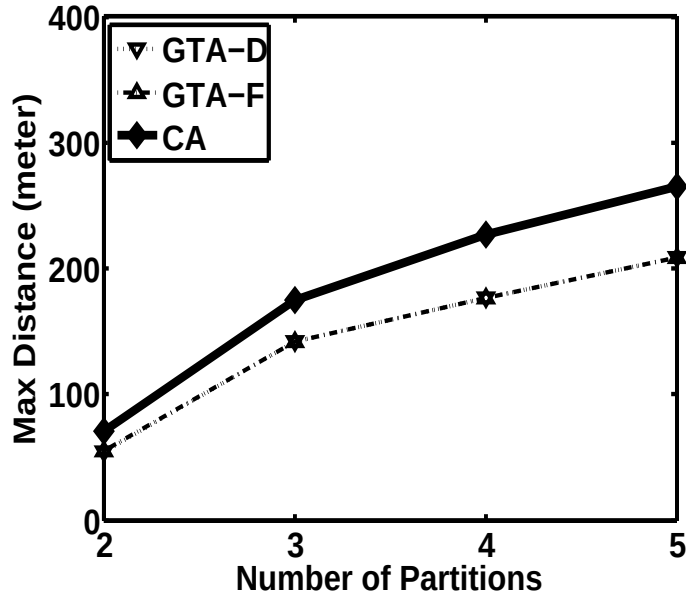


Figure 6.12: Maximum travel distance with varying number of partitions. The number of nodes is fixed to 150. Results in Figure 6.11 is magnified for *GTA* and *CA*.

at the center. This is due to the high node concentration at the center of the partitions when *GTA* is used. Deployment of the relays to the center reduces both the traveling cost and the number of relays due to the decreased distance between the initial and final locations. Thus, less relays will be sufficient to complete recovery for *GTA*.

The impact of the disaster scope on the approaches is provided in Figure 6.18. Again the cost will be same for *CA* for both of the deployment scenarios. As before, relay count for *CA* is less than that of *GTA* in any case. This is expected since *CA* strives to determine the minimum number of relays to connect partitions. *GTA* does not have such an objective and performs worse than *CA*. For *GTA*, again, deployment to the center declines the cost comparing with the deployment to the edge. This is due to closer distance to the final locations from the center of the application area.

6.5 SUMMARY

In this chapter, we presented distributed approaches for relay node placement problem in partitioned WSNs. The first approach utilizes repelling forces from Physics and stretches

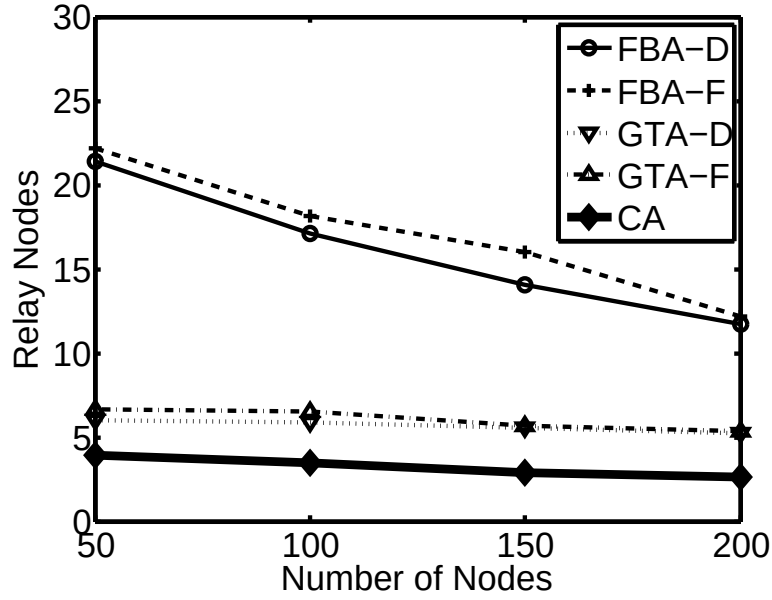


Figure 6.13: The number of relays deployed with varying number of nodes. The number of partitions is fixed to 2.

the initial network until the connectivity is ensured. The second approach is designed to reduce the movement cost using a Game Theoretic approach. This approach utilizes the degree pdfs of the partitions based on the use of primitive cameras on the relays. The approach strives to minimize not only the number of relay nodes to be deployed but also the total movement cost of the relays. Based on the pdf of the partitions, relay nodes determine the target partition to connect to. The partition will have the same Nash equilibrium once connected with the relay nodes. Recovery is completed when the connectivity with all the partitions is restored and a system-wide Nash equilibrium is reached.

The experiments have shown the superiority of *GTA* with respect to *FBA* under all conditions considered. In addition, compared to a centralized approach, *GTA* can provide slightly better travel distances despite using more relays. The results also proved the scalability of *GTA* regardless of the relay deployment scenario used. There is a tradeoff between deployment of additional relays and the movement cost. While more relays can decrease the movement cost, this needs to be carefully planned based on the relay node hardware cost and the deployment cost. Finally, the experiments also revealed that the

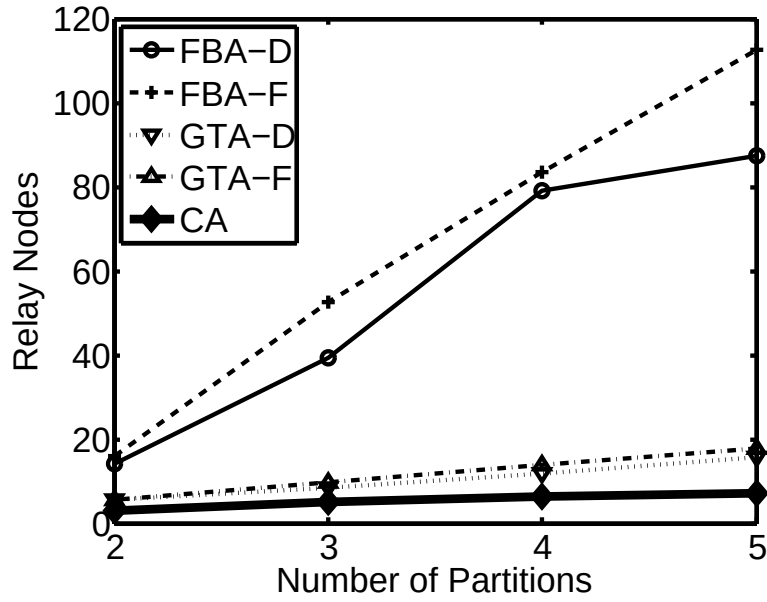


Figure 6.14: The number of relays deployed with varying number of partitions. The number of nodes is fixed to 150.

initial deployment location affects the performance. Deployment at the center is better for *GTA* while this is not the case for *CA*. As a future work, we plan to consider obstacles and terrain characteristics as well as the residual energy of the nodes in movement decisions. In addition, we would like to implement *GTA* in a small testbed of mobile sensors.

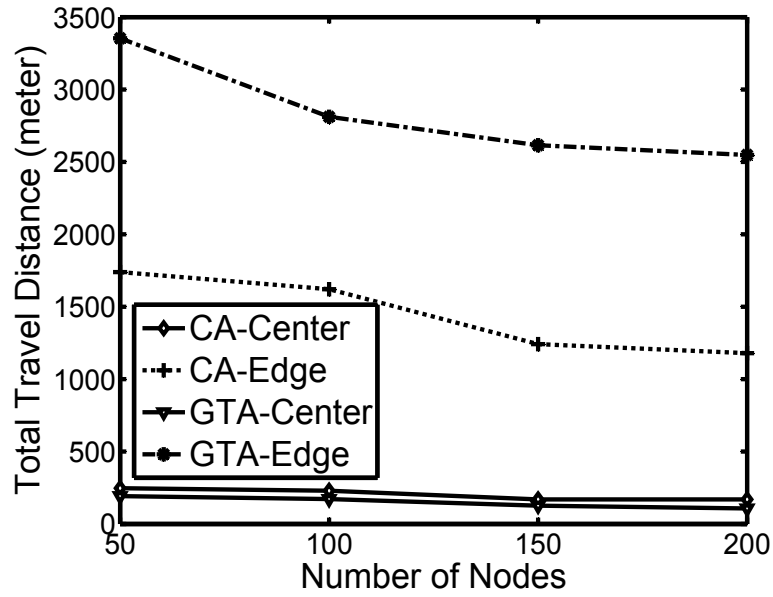


Figure 6.15: Total travel distance with varying number of nodes. The number of partitions is fixed to 2.

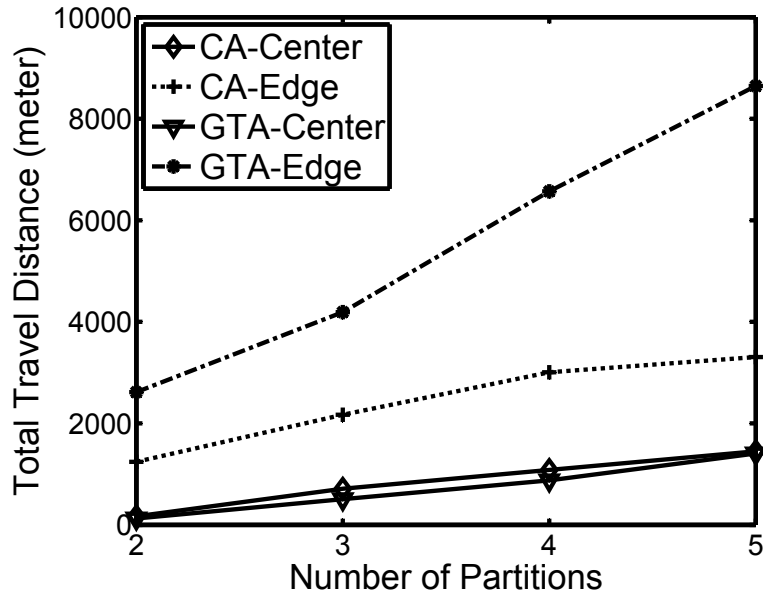


Figure 6.16: Total travel distance with varying number of partitions. The number of nodes is fixed to 150.

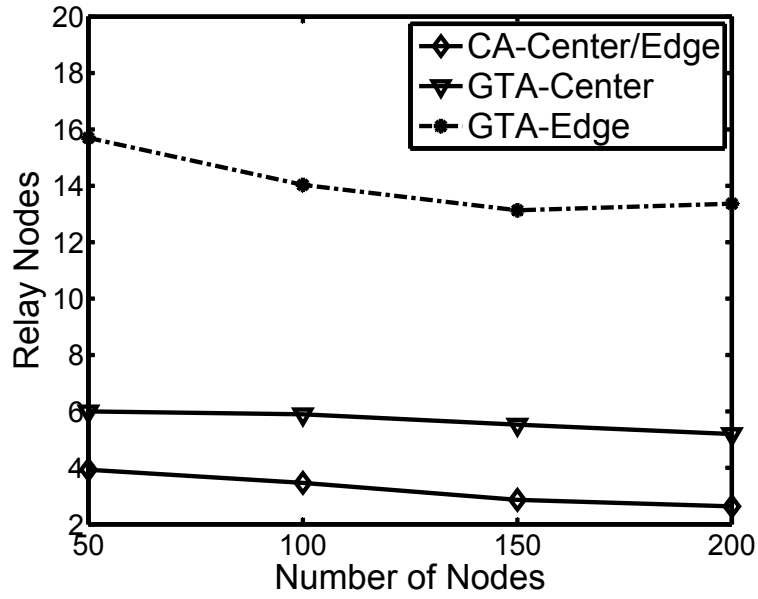


Figure 6.17: The number of relays deployed with varying number of nodes. The number of partitions is fixed to 2.

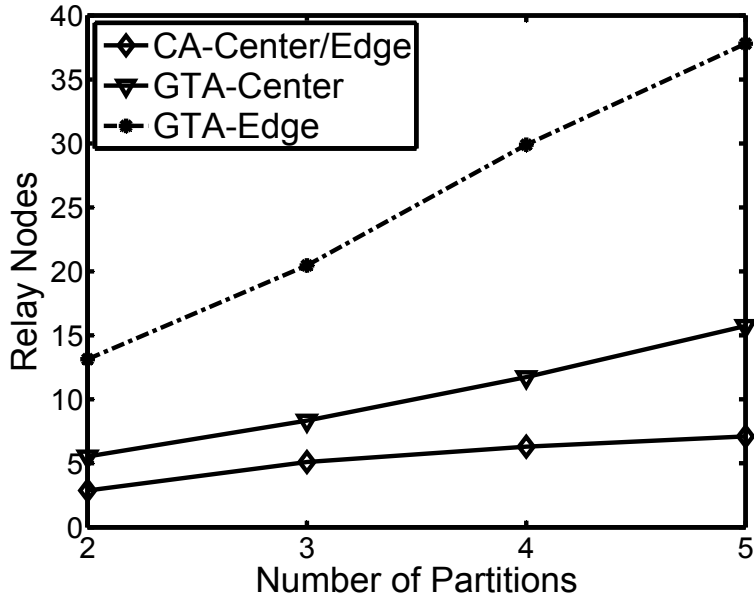


Figure 6.18: The number of relays deployed with varying number of partitions. The number of nodes is fixed to 150.

CHAPTER 7

CONNECTIVITY RESTORATION IN DISJOINT WIRELESS SENSOR NETWORKS USING LIMITED NUMBER OF MOBILE RELAYS

In this chapter, we present an approach to restore network connectivity through placing additional relay nodes in the damaged areas. In some cases there may not be enough relays to reconnect all the partitions with the sink node. In such a case, some of the relays can exploit their motion capabilities and temporarily act as a mobile data collector (MDC) between partitions providing intermittent connectivity for the nodes sitting in those partitions. Nonetheless, due to increased data latency intermittent connectivity creates, the number of such MDCs need to be minimized. On the other hand, given that the energy resources for an MDC is limited, an upper bound on the travel distance overhead for an MDC needs to be imposed. This chapter proposes a relay placement algorithm which guarantees connectivity by maximizing the number of stable connections while meeting the maximum tour constraint on the MDCs. The approach first determines the number and location of relays to restore connectivity by establishing stable links using a Steiner Minimum Tree (SMT) heuristic. Assuming that the number of available relays is less than the needed count, the algorithm determines how many of the available relays need to be stationary and how many of them should act as MDCs. By initially assuming all relays as MDCs, an iterative procedure is followed to reduce the MDC count while meeting the maximum tour length constraint. Specifically, groups of partitions are created and assigned to MDCs for touring.

In this chapter, we propose an algorithm to guarantee the connectivity of a partitioned WSN with a given number of mobile RNs. The connectivity is assumed to be from every partition to the sink partition (i.e., the partition where sink exists). The algorithm is greedy in the sense that it strives to maximize the number of sensors that are connected with the sink partition by first linking the partitions that are closer to the sink partition and larger

in sensor count. The linking is done via RN placement by using one of the traditional relay placement algorithms such as [43].

Given that there is a tour length constraint, the algorithm starts by assuming that all the RNs are used as MDCs. To this end, the partitions are clustered based on the number of RNs and an MDC is assigned to each cluster. In clustering, interface sensors are used to represent the partitions. The clustering stage is crucial for balancing the tour length among different MDCs so that the maximum tour length is not exceeded. The algorithm then starts iterating to reduce the MDC count and hence increases the number of stable links. At each iteration, a partition is linked with the sink partition by employing MDCs as stationary RNs and the maximum tour length is checked with the new partitions. If the tour length constraint is not exceeded, the process continues. Otherwise, the algorithm stops and reverts to the previous iteration in the worst case to guarantee a solution. Note that the tours for MDCs are computed based on the traveling salesman problem (TSP) which is NP-hard [59].

The evaluation of the proposed approach under various tour lengths and RN count reveals the efficiency of the approach in terms of connected sensors to the sink. In addition, the number of available RNs and tour bound significantly affect the maximum/total length of the tours. Compared to the approach in [36], the simulation results indicate that the tour lengths are significantly smaller due to balanced clustering when all the RNs act as MDCs.

This chapter is organized as follows: Assumptions and the problem definition are given in Section 7.1. The approach is presented in Section 7.2. The validation experiments and simulation results are provided in Section 7.3 and the chapter is concluded in Section 7.4.

7.1 PRELIMINARIES

7.1.1 Assumptions

We assume a partitioned WSN with a single sink node which is formed due to a massive damage or randomness of the initial deployment. A partition is defined as an isolated but connected network of nodes and the deployment region is assumed to have multiple of these partitions. The RNs are assumed to be mobile and can move freely to collect data from different partitions. For each partition, there can be an interface sensor which collects the data from the partition and transmits to a mobile RN known as MDC when MDC visits the partition.

7.1.2 Problem Definition

Given a set of n nodes which form $k \geq 2$ disjoint partitions, our goal is to provide a solution to restore network connectivity by introducing $m > 0$ mobile RNs $\in R$ to the system. Mobile RNs can either be used to form stationary connection between partitions or be employed as MDCs that provide intermittent links between partitions. Mobile RNs in the first group comprises the set MDC while the mobile RNs in the second group is represented by the set SR . $t = |MDC|$, $s = |SRN|$ and $m = t + s$. Let P_{per} and P_{int} be the sets of partitions connected with a stationary connection and through MDCs respectively. We also define a sub-goal: While restoring connectivity, maximizing the number of nodes connected to the sink with a stationary connection (i.e., maximize $\sum_{i=1}^p s_i$ where $p \leq k$, $p = |P_{per}|$ and s_i is number of nodes in $partition_i$). Finally, we define a constraint as follows: The tour length for an individual MDC should not exceed a certain threshold i.e., $d_i \leq L$ where d_i is the tour length of MDC_i and L is the maximum allowed tour length.

Normally, if we assume unlimited number of RNs, the problem can be modeled as SMT problem and one of the heuristics in the literature can be used to determine the locations and count for the RNs. Since, each partition is represented by sensors, we utilize the CIST approach in [43] which returns not only the count and locations of RNs but also

the interface sensors to be used as seen in Fig. 7.1. Let's assume that the minimum number of RNs needed to restore connectivity is C (i.e., as a result of running CIST) then we only consider the case where $C > m$ in this chapter. For the case $C < m$, a stable connectivity can be restored for all the partitions. Next, we detail our approach for handling the former cases.

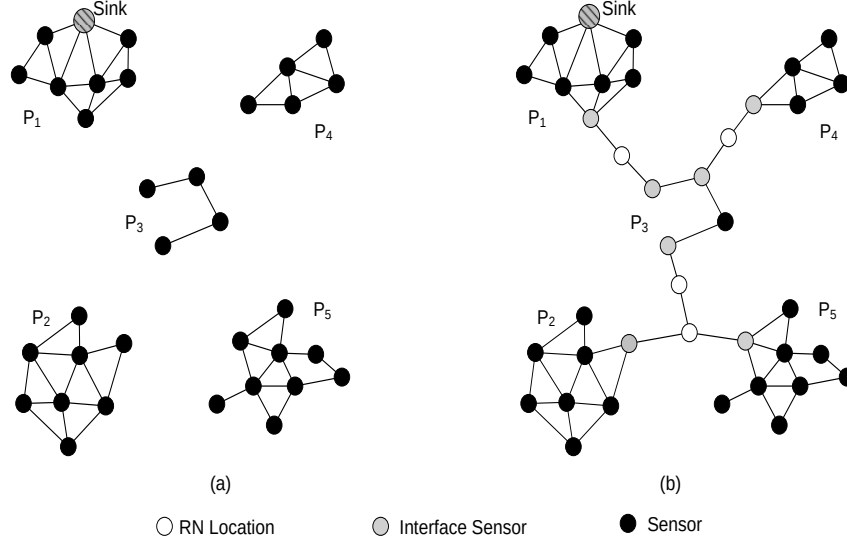


Figure 7.1: A sample illustration for a partitioned WSN (a) and the planned RN locations that will reconnect the WSN as a result of CIST [43] algorithm (b)

7.2 APPROACH

7.2.1 Overview

We propose an approach which works in 2 phases. As mentioned before, in the first phase, SMT of the network is obtained through CIST [43] algorithm. As part of SMT, we obtain the locations for placing the RNs. CIST uses partition representatives while determining SMT and thus part of the output will be the IDs and locations of these representative sensors. In the second phase, we identify the partition(s) to be connected with the sink if possible and the number of RNs to be served as MDCs (t) as well as the tours of MDCs. The second phase has a clustering component which identifies the clusters, the tours to visit all the partitions in that cluster and checks if the maximum tour length

has been violated or not. The tours are computed using Traveling Salesman Problem (TSP) [59] as done in [39]. Next, we detail our algorithm for the second phase which is our main contribution in this chapter.

7.2.2 Iterative Second Phase

In this phase, we strive to provide stationary links to as many partitions as possible by placing the RNs to the SMT locations. However, we need to make sure that there is at least one RN which can be used for MDC and can provide a tour of length less than the maximum, L . Therefore, we follow an iterative approach where initially all the RNs are assumed to be MDCs to be touring among the partitions. As a result, each MDC is assigned to a tour based on determining a TSP among the partitions. These tours are formed using a clustering algorithm which considers a set of partitions within a certain cluster. Obviously, if the length of the maximum of these tours exceeds L , then there is no solution for our problem. Otherwise, we start linking partitions with the sink partition by placing some of the RNs, meaning that we reduce the MDC count iteratively to increase the number of stable links. Next, we explain the details of clustering and tour length checking.

7.2.2.1 Clustering

As input to our clustering algorithm, we use the partition representatives (i.e., interface sensors) from each partition and the created SMT links as a result of running CIST. Since m RNs are available, we create m clusters (i.e., one MDC per cluster). The clustering is designed in such a way that the tour lengths in all clusters are balanced and have similar costs with each other. This is because we would like to minimize the maximum tour length so that it does not exceed the given threshold L . In this way, a feasible solution for the general problem will be available while the number of stable links will be increased.

The idea is similar to that of designing a Huffman coding tree. We initially consider each link of the SMT a cluster (i.e., with two partitions). Then we sort them based on the

length of the link which is the number of RNs to be deployed on that link. Starting from the smallest ones, we pick two links that share a partition and merge them to create a new cluster having three partitions. Among all possible combinations, the one which creates the smallest tour with three partitions is picked by the algorithm. In this way, merging is continued until the number of clusters match with m . An example is provided in Fig. 7.2.

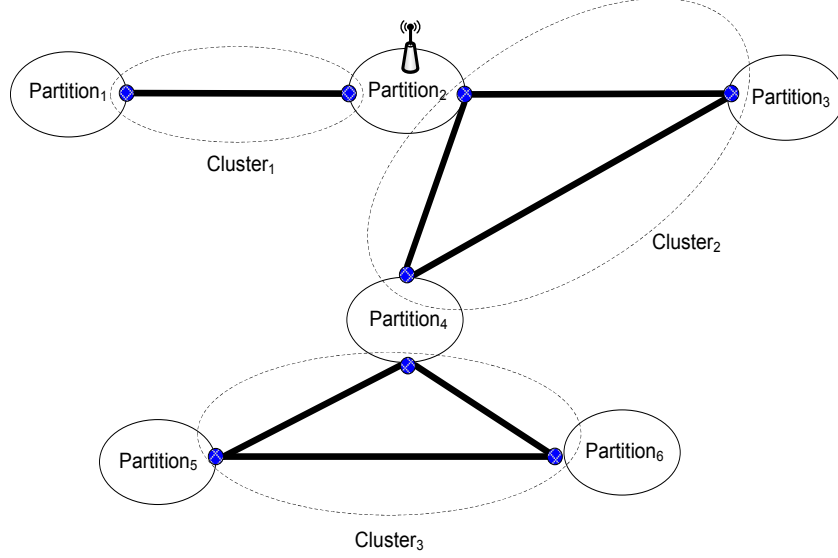


Figure 7.2: The creation of clusters based on the number of RNs. Note that a partition may have multiple interface sensors and our approach considers them as separate locations as opposed to previous work. In the case of *partition₄*, we assume that one of the interface sensors can relay all the data it collects via routing to the other interface sensor so that the data can be forwarded to the sink.

7.2.2.2 Tour Bound Checking and Re-clustering

After clustering, the approach starts iterations and reduces MDC count at each iteration until the used MDCs violates the required tour bound. This is done as follows: First, we determine how many of the RNs will be used to link a partition to the sink partition. Since connectivity is with the sink node, the goal is to link one of the largest and possibly closest partition with the sink node by populating the RNs to the SMT locations between the sink partition and the determined partition. To determine the partition to be linked, we define a cost function based on the number of RNs to be used to connect that partition

with the sink node. For instance, if Par_x serves Par_y to connect the sink partition, then $cost_{par_y-sink} = cost_{par_y-par_x} + cost_{par_x-sink}$. Once all the partitions are associated with a cost, the one with the minimum cost is selected to be connected with the sink partition by a stationary link. Then, the partitions and the number of available RNs are updated accordingly (i.e., MDC count is reduced to say, m').

Once the updates are done, it is time for a next iteration in which the same process continues. Basically, m' MDCs are used to create clusters and tour lengths are computed. If the maximum tour length is less than L , we continue the same process but also commit the previous iteration. Otherwise, we stop and come back to the committed solution in the worst case. Specifically, if c RNs were placed in the previous iteration, one of those RNs (the one closest to the neighboring partition) is removed from the stationary link it formed. Then the tour bound is re-checked. In the worst case, $c - 1$ RNs will be removed as we already know that the case without c RNs has already been committed and can guarantee a result. An example for the execution of the algorithm is given in Fig. 7.3. The pseudo code of the algorithm is skipped due to space constraints.

7.2.3 Determining Tour Path

At each iteration, the tours are computed already. However, with the last iteration some additional RNs may exist and thus new tours are computed. The computation of a tour is done based on determining the convex hulls of the representative nodes in each partition. The same algorithm of [39] is followed except using different interface points for each partition. Some examples of the tour computation is provided in Fig. 7.4. The details are skipped due to space constraints.

7.2.4 Time Complexity Analysis

For a network with n nodes, E edges connecting these nodes, and k partitions, the cost of the CIST algorithm is $O(((E + n \log n) \times k^2) + k^3)$. Tours can be determined while

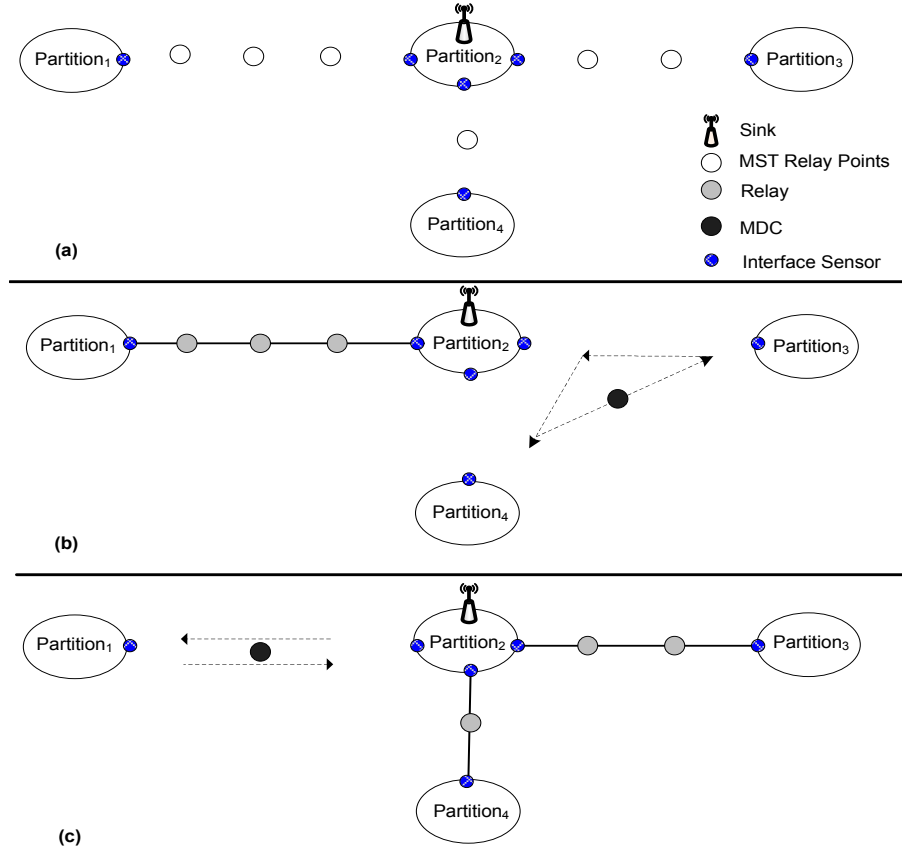


Figure 7.3: (a) A disjoint WSN with MST points to be filled in. (b) A solution with 3 RNs and 1 MDC. (c) Another solution with 3 RNs and 1 MDC. The tour length in this case is less than that of (b) even though the same number of RNs are used in total.

partitions are clustered with a complexity of $O(p^2 \times m)$, for p partition representatives in total and m MDCs. In the worst case, clustering and tour computation will be performed v times for v SMT points during the revocation of the last iteration (i.e., when the maximum tour length L is exceeded). Since $v < m$, computational complexity of the algorithm can be expressed as $O(((E + n \log n) \times k^2) + k^3 + p^2 \times m^2)$ or simply as $O(E + n \log n)$ for a network with limited number of partitions and MDCs assuming $p^2 < n \log n$.

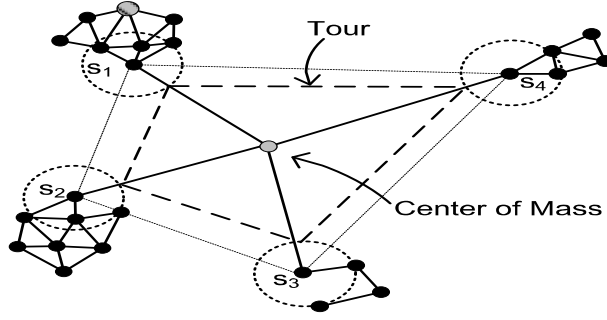


Figure 7.4: Tour computation for 4 partitions with an interface sensor. The MDC will come close to the transmission range of each sensor to collect data.

7.3 EXPERIMENTAL EVALUATION

7.3.1 Experiment Setup

The effectiveness of the proposed approach is validated via simulation. Disjoint WSN topologies are considered in a $800 \times 800\text{m}$ area. The transmission range for sensors is set to 50m and the number of nodes and partitions are varied. Each test case has been tested with 30 different topologies for significance and the average is reported.

Two performance metrics are considered: 1) Total number of connected sensors to the sink which indicates the stability of the links; 2) Maximum/total tour length which hints on the data delay and energy efficiency. We compared our approach with IDM-kMDC [36] in terms of maximum and total tour lengths when all the RNs are acting as MDCs in our approach. Recall that IDM-kMDC assumes all the RNs as mobile (i.e., MDCs) and computes tours for all the MDCs.

7.3.2 Evaluation and Analysis

7.3.2.1 Number of Sensors

One of the goals of our approach is providing stationary links to as many sensors as possible without exceeding the bound L for MDCs. To this end, we conducted experiments by first keeping the number of RNs fixed as $m = C/2$ where C represents the total number of RNs to restore connectivity and varying L from 400m to 700m. The results are depicted

in Fig. 7.5. It can be observed from 7.5 that the number of sensors served with stationary links is increasing with the increased bound L . This is expected since with the allowance of longer tour lengths, our approach provides stationary links to more sensors leaving less MDCs for touring. On the other hand, the number of sensors connected with the sink declines when the number of partitions is increased. This is due to the distribution of the sensors into more partitions and the increased need for recovery. Even if some partitions are connected, since they include fewer nodes, the total number of connected sensors declines.

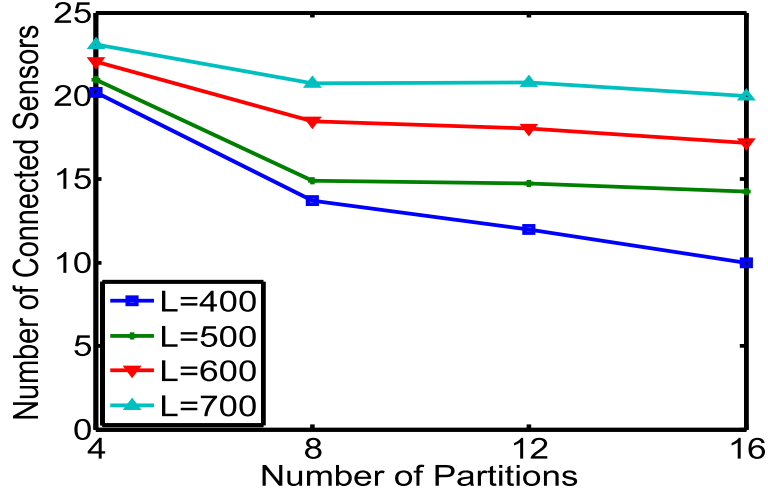


Figure 7.5: # of connected sensors for varying partition count and L .

We also conducted another experiment where we kept the bound fixed as $L = 500$ and varied m in the network from $C/2 - 1$ to $C/2 + 2$. The number of partitions is also varied from 4 to 16 as before. The results are represented in Fig. 7.6. It can be observed from Fig. 7.6 that the number of sensors connected with the sink increases with the increased m . This is attributed to the fact that there are more RNs in the area and some of them are used to link more partitions. On the other hand, the number of sensors connected with the sink declines with the increased number of partitions because of the same reason mentioned above.

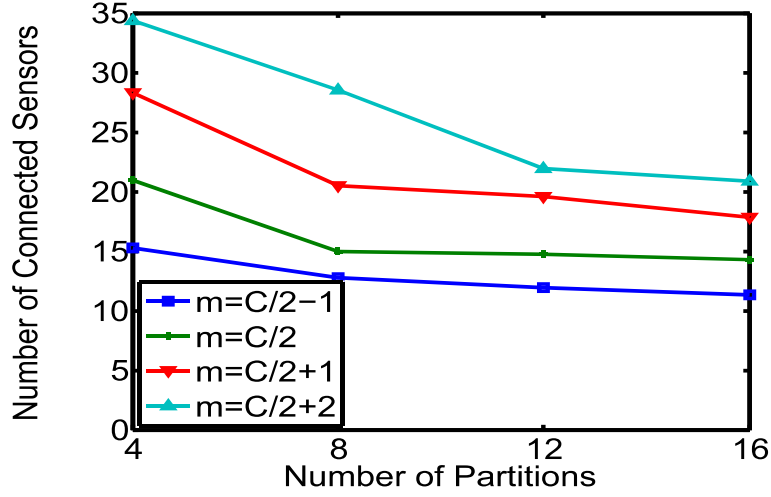


Figure 7.6: # of connected sensors for varying partition count and m .

7.3.2.2 Maximum Tour Length

To evaluate the proposed approach in terms of the maximum tour length incurred, we conducted two experiments. In the first one, m is kept constant as $m = C/2$ and tour bound L is varied from 400m to 700m. The number of partitions in the network is also varied from 4 to 16. The results are represented in Fig. 7.7. It can be observed from Fig. 7.7 that maximum tour length is increasing when the tour bound L is increased. This is expected, since with the increased upper bound, less MDCs are used to complete longer tours so that more RNs can be used as stationary to connect more sensors to the sink.

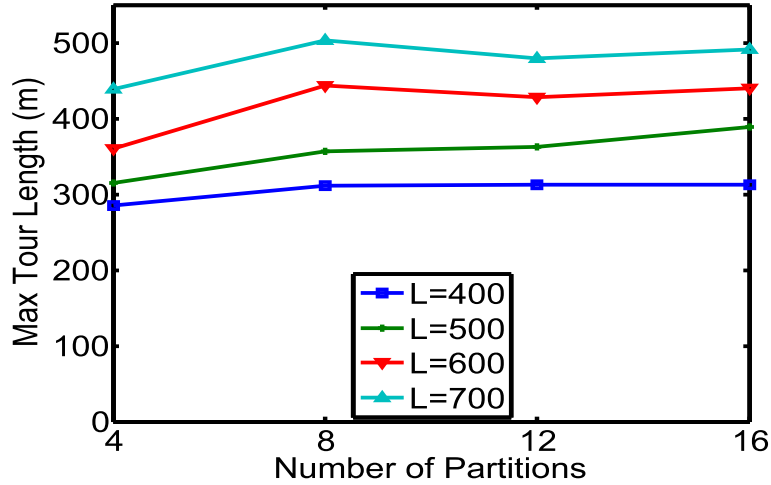


Figure 7.7: Maximum tour length for varying partition count and L .

In the second experiment, we kept the bound fixed as $L = 500$ and vary m in the

network from $C/2 - 1$ to $C/2 + 2$. The results are shown in Fig. 7.8. It can be observed from Fig. 7.8 that maximum tour length declines for increasing number of RNs when 4 partitions are present in the network. This is because extra RNs can be used to shorten the tours of MDCs. However, for increased number of partitions, the pattern is not very similar to this case. When the number of partitions is increased, with a higher chance, there will be partitions requiring less RNs to establish a stationary link (i.e., the area and the number of sensors are fixed, so partitions become closer to each other). In the favor of establishing a stationary link, the extra RNs will be deployed and thus for the remaining RNs which will act as MDCs, the algorithm will extend the tour lengths.

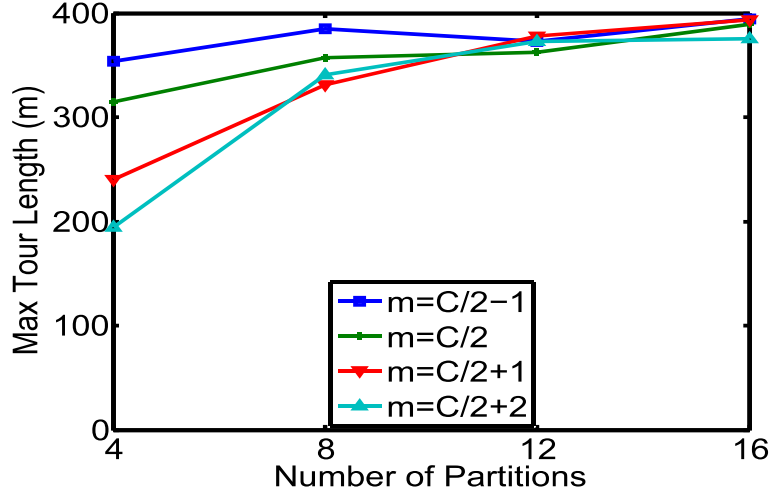


Figure 7.8: Maximum tour length for varying partition count and m .

7.3.2.3 Comparison with IDM-kMDC

The proposed approach was also compared with IDM-kMDC [36] in terms of maximum and total tour lengths. For this purpose, we picked several topologies with varying partition count and m where our approach produced a result. IDM-kMDC was run using the same topologies and same m values. A sensor which sits in the center of partition is picked as the partition representative. The results for maximum and total tour lengths are shown in Fig. 7.9 and Fig. 7.10 respectively. The topologies are indicated with topology IDs on x coordinate. As can be seen from the figures, our approach significantly reduces the

maximum and total tour lengths since it provides a load balanced clustering where the tour lengths of different clusters do not deviate much from each other. This eventually reduces the maximum tour length.

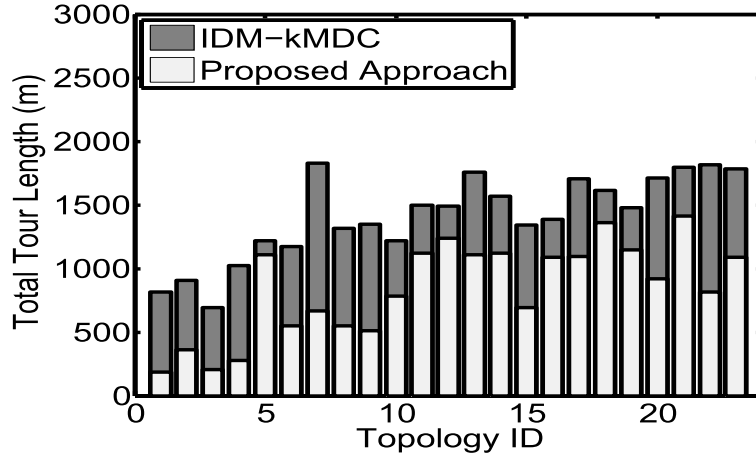


Figure 7.9: Total tour length for different topologies.

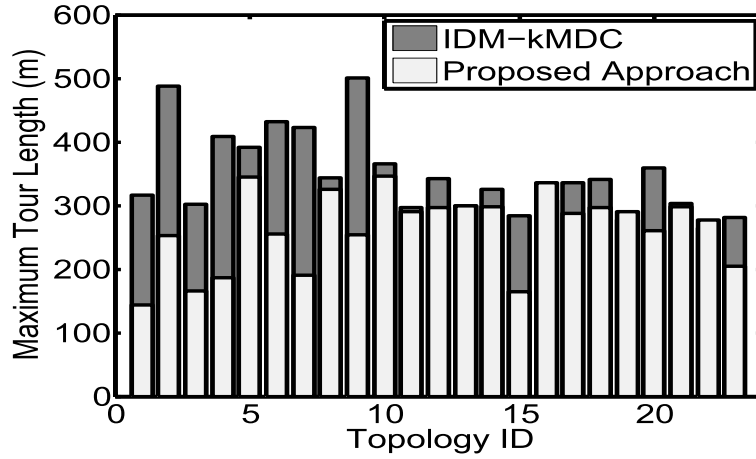


Figure 7.10: Maximum tour length for different topologies.

7.4 SUMMARY

In this chapter, we have proposed relay placement approach to provide intermittent connectivity when the number of RNs is not sufficient to restore the connectivity with stable links. The approach picks some of the RNs as MDCs so that they can serve multiple partitions and collect their data. The tour length is restricted in order to provide energy efficiency for the MDCs. The approach iteratively reduces MDC count by starting with

all RNs as MDCs. This is done to check the maximum tour length at each iteration. The process stops when the maximum tour length is exceeded. Comparing with the previous approach IDM-kMDC, our approach can provide a better load balance among the tours and thus reduces the maximum and total tour lengths. This eventually provides more RNs to be deployed as stationary and hence increases the number of connected sensors.

CHAPTER 8

MOBILE DATA COLLECTOR ASSIGNMENT AND SCHEDULING FOR MINIMIZING DATA DELAY IN PARTITIONED WIRELESS SENSOR NETWORKS

In this chapter, a QoS aware solution is presented to avoid rendezvous waiting delays occurred due to dependability of other MDCs to forward data. Typically, an MDC serves multiple partitions to collect their sensed data and relay it to the sink node if the sink is in one of the served partitions. If not, then the data needs to be passed to another MDC serving the sink partition at a certain rendezvous location. This causes an extra waiting time for the MDCs that reach the meeting points before others. In this chapter, we propose a scheduling algorithm to reduce the rendezvous waiting delay based on the tour lengths. To completely eliminate this waiting delay, we also present a solution with multiple sinks. In this approach, the partitions of the WSN are clustered using p -center optimization and Voronoi cells. An MDC is then assigned for touring each cluster's partitions. In this way, the dependency of MDCs to relay their data to other MDCs is eliminated and the tour lengths of MDCs are balanced which minimizes the end-to-end delay significantly.

Recently, several research challenges regarding the exploitation of MDCs have been tackled. The main focus of the studies was to reduce the tour lengths of MDCs when they are visiting each partition to collect the sensor data. Since, typically multiple MDCs are available, multiple tours exist and the partitions involved in these tours are pre-arranged based on a clustering algorithm [39][36]. To have load-balanced tour lengths (i.e., to reduce the maximum tour length and data latency), the existing clustering algorithms strived to provide balanced clusters [39][60]. In such clusters, MDCs collect data from all of the partitions and then relay it to the sink node, if in that partition, or to another MDC so that it will relay data to the sink node. This introduces the problem of rendezvous waiting which has not been considered in the previous studies. The implicit assumption in such

studies is that the sensors within the partition can hold the data collected from an MDC indefinitely and then forward it to another MDC once it is nearby. Since sensors have limited buffers, this may not be feasible when multiple MDCs are dumping their data to the same sensors. Therefore, clustering should be done by also considering the impact of possible rendezvous waiting so as to minimize the overall data latency.

In this chapter, we first demonstrate the impact of clustering on data delay with a single sink. We introduce rendezvous waiting delay as one of the major factors when determining the total data delay when several partitions exist. We then propose a scheduling algorithm to alleviate the waiting time by checking the possible locations which have potential to introduce rendezvous delay. Specifically, the waiting is eliminated if another tour is possible within the rendezvous waiting time. In this way, some of the partitions' data will be transferred without waiting for the data coming from other partitions which will minimize the average delay in the network.

Since this scheduling does not eliminate the rendezvous waiting completely, we later introduce a clustering approach which eventually eliminates the waiting time completely. The idea is to designate multiple sinks to collect data to eliminate the dependency of an MDC on other MDCs to relay packets. This raises the problem of load-balanced clustering of partitions based on a given number of sinks. As part of this problem, we tackle two sub-problems: First, determining the right locations for the sinks among the partitions; second, creating the clusters of partitions around these sinks.

For determining the sink locations, we follow the idea that the locations should be apart from each other as much as possible as used in p -center optimization problem [61]. Note that during this process each partition is assumed to be represented as a single entity whose location is the centroid of all the sensors within that partition. Once the sink locations are determined, we assign the sensor nodes that are closest to these locations as the sinks to collect data from its cluster.

For clustering, we propose an approach based on Voronoi cells. We define an area of

proximity (Voronoi cell) for the sinks where the partitions within the Voroni cell will have the minimum distance to the corresponding sink compared to other sinks. Once the clusters are formed, each cluster is served with an MDC to collect the data from the partitions in the cluster and deliver to the sink affiliated with that cluster. We use Traveling Salesman Problem (TSP)-based modeling while planning the tours [39] for the MDCs.

The proposed approaches are evaluated under a variety of conditions via extensive simulations. The simulation results indicate that end-to-end delay decreases with our proposed scheduling approach. The delay is further reduced upon applying the proposed clustering schemes.

This chapter is organized as follows: Assumptions and the problem definition are given in Section 7.1. The approach is presented in Section 7.2. The validation experiments and simulation results are provided in Section 7.3 and the chapter is concluded in Section 7.4.

8.1 PRELIMINARIES

8.1.1 Assumptions

We assume a WSN with disjoint partitions detached from the rest of the network due to initial deployment or node failures. Initially, we assume a single sink which collects data from all the sensors within this WSN. The nodes in the WSN are assumed to be stationary and therefore the network topology cannot be restructured without any external intervention. We assume the availability of mobile RN(s) referred to as MDCs equipped with a memory large enough to store the data of several sensor nodes. We also assume the application area is obstacle-free.

Connectivity of a partitioned network can be restored by deploying RNs to locations identified by an RN placement algorithm. However, excessive number of RNs may be required to guarantee connectivity and we assume the lack of sufficient RNs available. Thus, we employ RNs as MDCs to collect data from partitions and deliver to sink so that an intermittent connection is provided to ensure the sustainability of data communication.

We use an RN placement algorithm, CIST [43], to determine the minimum RN count to guarantee connectivity and the interface sensors for the partitions (i.e., the sensors which will communicate with MDCs). Based on the minimum number, we assume MDC count (i.e., it should be less than this number). A sample demonstration of the CIST showing the interface sensors is illustrated in Fig 7.1.

8.1.2 Problem Definition

Our problem can be more formally defined as follows: “Given a set of n nodes $\in N$ which form $k \geq 2$ disjoint partitions, our goal is to restore network connectivity by employing $m > 0$ (mobile) RNs in the area while minimizing the average data delay from sensors to the sink.”

The data delay is defined as follows: Once the connectivity is restored, each node, n_i , will transmit its packets to the destination (sink) through the upstream path P_i . P_i comprises a set of node IDs representing hops to reach the sink. Thus, for a delay, d_j , incurred at each hop, total delay of a packet sent from n_i to sink can be stated as $\sum_{j=1}^{|P_i|} d_j$ where $|P_i|$ denotes the number of hops in P_i .

As part of d_j , we consider three different types of delay where a packet may be subject to. If the hop j is not served with an MDC, delay can be simply defined as: $d_j = R_j$ where R_j denotes the transmission delay. On the other hand, if the hop j is served with an MDC RN_j , then delay on hop j can be stated as $d_j = T_j + W_j + R_j$ where T_j denotes the touring delay of RN_j and W_j denotes the waiting delay of the immediate MDCs exchanging data with RN_j .

As part of the solution, we investigate two approaches: First, we will assume a single sink node and propose a scheduling algorithm to minimize the waiting delay. Second, we will assume multiple sinks and propose a clustering algorithm to further reduce the data latency.

8.2 RENDEZVOUS SCHEDULING OF MDCS FOR REDUCED DELAY

8.2.1 Background on MDC Tours

In this section, we briefly explain how the partitions are clustered and the tours are computed when the network has a single sink. Depending on the number of available MDCs, say m , we create m clusters of partitions and assign each cluster an MDC. The clustering starts with pairing of partitions and then merging them until the number of clusters matches with m [60]. The MDC is expected to tour the partitions within a cluster and collect data from interface sensors. The tour of the MDC is computed using Traveling Salesman Problem (TSP) [59] as done in [39]. An example of clustering and tours is provided in Fig. 7.2.

Since each MDC collects data from all the partitions within a cluster, it needs to forward them to the sink node. If the sink is not in one of the partitions within the cluster, then the MDC should forward the collected data to another MDC (e.g., MDC of *cluster*₃ should forward data to MDC of *cluster*₂ that has the sink). Since the MDC may not dump all its collected data to a sensor in the partition due to buffer constraints of the sensors, this may introduce some rendezvous waiting time which has not been considered in the previous work. Especially in WSNs with several clusters, waiting time for MDCs can be a dominating factor in the total delay.

Next, we analyze this delay as part of the whole data delay and then propose a scheduling algorithm to reduce it.

8.2.2 End-to-end Delay with a Single Sink

Given that rendezvous waiting can be a big issue impacting the delay, in this subsection we will describe a computation mechanism for determining the waiting time of MDCs and assess its impact on the overall delay.

Each MDC will wait at certain meeting points for the MDCs touring in the neighboring clusters. This is referred to as “wait-for-all children” approach to synchronize all the

data going to the sink node. A meeting point resides at the intersection of two or more clusters. Since the whole set of partitions will be connected, clusters will intersect at certain partitions. For instance, if two clusters contain sensor nodes from the same partition, then the tours will intersect at the meeting points which represent the locations to contact the interface sensors of the common partition. Thus, meeting point does not represent a point where two MDCs reside at the same time, but rather two different positions possibly apart from each other with a distance larger than their transmission range (see Fig. 8.1). The waiting time of an MDC at a meeting point depends on the immediate predecessor MDCs (i.e., children) and the immediate successor MDC (i.e., the MDC that is closer to sink). Multiple MDCs can constitute the set of children but only one MDC will be the successor.

A sample scenario depicting meeting points and children is represented in Fig. 8.1. In this example, partition P_4 is the common partition of clusters $C_a \cap C_b \cap C_c \cap C_d$. Based on the direction towards the sink, it can be observed that MDC_a , MDC_b and MDC_c are the predecessor (children) MDCs of MDC_d . On the other hand, partition P_5 is the common partition of clusters $C_d \cap C_e$ and MDC_e is the successor MDC of MDC_d . Thus, we can identify two meeting points for MDC_d , one for P_4 and one for P_5 . The interface nodes are already obtained during clustering and the locations to contact P_4 and P_5 are determined while forming the tour of MDC_d . Now we detail how to calculate the rendezvous waiting time of MDC_d at meeting points m_2 and m_3 .

In order to initiate the data exchange at m_2 , MDC_a , MDC_b and MDC_c should be present on m_1 while MDC_d is present on m_2 . They can then forward their data via the sensors in the partition to MDC_d . For the waiting time of MDC_d at m_2 , in the worst case, we can assume that MDC_d will have to wait the children with the longest tour length (e.g., MDC_a) to finish a complete tour. Similarly, in order to initiate the data exchange at m_3 , MDC_e should be present at m_4 while MDC_d is at m_3 . For the waiting time of MDC_d at m_3 , in the worst case, we can assume that MDC_d will have to wait for MDC_e to finish a complete tour. Thus, the total waiting time of MDC_d at m_2 and m_3 can be expressed as

in Equation 8.1 where t is the number of predecessor MDCs.

$$W^{MDC_d} = W_{m_2}^{MDC_d} + W_{m_3}^{MDC_d} = \max_{i=1}^t T^{MDC_i} + T^{MDC_e} \quad (8.1)$$

Since none of the predecessor MDCs of MDC_d has a child, no waiting delay occurs for MDC_a , MDC_b and MDC_c . Otherwise, waiting time of these MDCs should also have been considered in a recursive way. Similarly, since MDC_e is connected to sink, no further waiting delay affects MDC_d at m_3 . Again, otherwise, MDC_d would be subject to waiting delays of MDC_e in a recursive way.

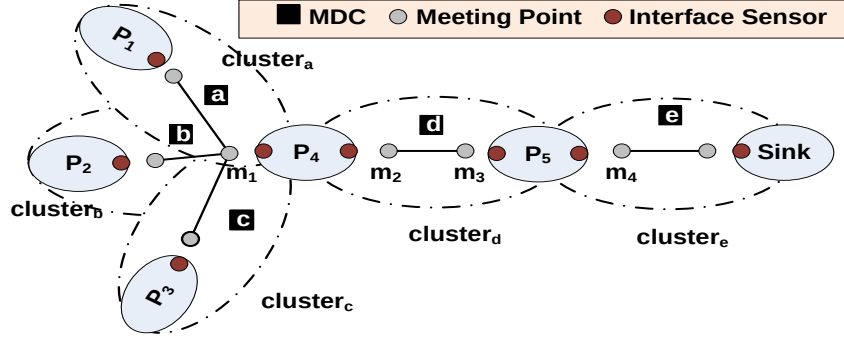


Figure 8.1: A sample illustration of the occurrence of the waiting time in the collaborating MDCs

8.2.3 Rendezvous Scheduling for Reduced Delay

In order to reduce the rendezvous waiting times, we strive to exploit the time period between these waiting times. Specifically, if the MDCs start moving at certain positions, waiting delay can be decreased.

We will explain the idea on an example. Let's revisit the sample topology in Fig. 8.1. Let the tour lengths of MDC_a , MDC_b , MDC_c , and MDC_d be $l + x$, $l + y$, $l - z$ and l respectively where $x, y, z < l$. Assume that the first data exchange is completed among MDCs at meeting points m_1 and m_2 . The next round for MDC_d will be after finishing a complete tour and returning back to m_2 . Due to its shorter tour length, MDC_c will be

present at m_1 but it will take another $\max(x, y)$ time for both MDC_a and MDC_b to also complete their tours. Therefore, MDC_c can check the feasibility of another tour rather than waiting there indefinitely for MDC_a and MDC_b . If $l - z$ length can be toured within the waiting time, MDC_c will prefer this tour to increase the data sampling rate from the sensors in P_3 .

Therefore, we can express the waiting delay for MDC_d as the summation of waiting delay at m_2 and m_3 as in Equation 8.2:

$$W^{MDC_d} = W_{m_2}^{MDC_d} + W_{m_3}^{MDC_d} = (\max - T^{MDC_d}) + (T^{MDC_e} - T^{MDC_d}) \quad (8.2)$$

where $\max = \max_{i=1}^t T^{MDC_i}$ and t is the number of preceding MDCs. Note that we take 0 as the waiting time if $\max < T^{MDC_d}$ or $T^{MDC_e} < T^{MDC_d}$. This is the difference from Equation 1 which may provide the opportunity to have 0 waiting time for some of the MDCs.

Once an MDC computes the waiting delays at each meeting point using the above equations (the algorithms for the computation are skipped due to space constraints), it can determine its schedule by comparing its touring delay with the waiting delays.

We propose a scheduling algorithm to initiate a tour if it has enough time to complete the tour before the corresponding waiting delay expires. It is important to identify the direction of the tour. If the direction is towards the sink, waiting delay of the successor MDCs should be considered. Otherwise, waiting delay of the predecessor MDCs should be considered. The pseudo-code for this proposed algorithm is given in Algorithm 9.

Specifically, in lines 1 and 2, waiting delay to be incurred due to the predecessor and the successor MDCs are computed. In line 3, the tour info of the MDC is obtained. The direction of the tour is determined in line 4. The tour may be towards the sink partition or the opposite direction. If the direction is towards the sink, in order to initiate another

Algorithm 9 schedule(MDC m)

```
1:  $sDelay = \text{getSuccessorWD}(m)$  ▷ get successor delay
2:  $pDelay = \text{getPredecessorWD}(m)$  ▷ get predecessor delay
3:  $t = \text{getTour}(m)$  ▷ get tour info of m
4: if  $\text{directionTowardsSink}(t) == \text{true}$  then
5:   if  $sDelay \geq TD(T) + pDelay$  then
6:      $\text{makeTour}(t)$ 
7:   end if
8: else
9:   if  $pDelay \geq TD(T) + sDelay$  then
10:     $\text{makeTour}(t)$ 
11:   end if
12: end if
```

tour, waiting delay to be incurred due to successor MDCs should be greater than the sum of touring delay and the delay to be incurred due to predecessor MDCs. This condition is checked in lines 4-7. In the opposite direction, waiting delay to be incurred due to predecessor MDCs should be greater than the sum of touring delay and the delay to be incurred due to successor MDCs to initiate another tour. This condition is checked in lines 8-12.

8.3 DELAY-AWARE CLUSTERING WITH MULTIPLE SINKS

8.3.1 Motivation

The phenomenon of rendezvous waiting emerges when the clusters are not formed in a star-like topology where the sink resides in the center. Such a clustering requires the deployment of the sink at the central area of the topology. However, we assume a random deployment where the sink can be located at any location. While it is possible to form all the tours to include the sink and hence avoid rendezvous waiting, this will cause to increase the tour lengths significantly which unnecessarily increases the touring delay. Moreover, as shown in the previous section, even though it is possible to perform scheduling to reduce rendezvous waiting time, it is not possible to completely eliminate it (as will be shown in the Experiments) with a single sink and thus it remains as a major factor of long end-to-end

delays.

Consequently, in this section, we present an alternative solution assuming multiple sinks are available in the WSN. Considering the fact that the waiting delay emerges due to dependency of MDCs on other MDCs, we provide an algorithm to cluster the partitions in such a way that the data of different clusters will be delivered to sinks simultaneously without any interactions with MDCs of other clusters.

8.3.2 Approach Overview

The approach eventually clusters the partitions but it first needs to determine the sink locations. Based on the available MDC count, sink locations are identified. For m MDCs, m sinks are selected in the network. Considering the location of the selected sinks, the network is clustered into m clusters. The data of the nodes in each cluster is collected by the sink of the cluster. An MDC is assigned for the cluster so that the partitions in the cluster are toured by this MDC. Since the MDCs are only responsible from their clusters, demand for data exchange between the MDCs of different clusters is avoided.

8.3.2.1 Determining Sink Locations

The goal in sink selection is to minimize the maximum distance from partitions to the sinks. This can be modeled as a p -center problem where one wants to build p warehouses in different cities. Given that there are n cities, the goal is to minimize the maximum distance between the cities and the warehouses. In our case, the warehouses are the sinks and the cities are the partitions.

Following this idea, the first sink is located based on the maximum distance to the rest of the partitions in the network. The distance to partitions is based on the centroid location (i.e., location of average x and y in the partition) of the partition. The first sink is added to the set of sinks. We determine the rest of the sinks based on the maximum distance to this set of sinks. Once the locations are identified, the closest sensors to these locations

are designated as sinks. After identifying the sinks, we proceed with the clustering of the network.

8.3.2.2 Clustering

Our proposed clustering approach exploits Voronoi cells while determining the partitions of each cluster. A Voronoi cell keeps the points whose distance to cell center is less than or equal to its distance to any other center in the region [62]. Based on the given sink locations, the network is divided into Voronoi cells as shown in Fig. 8.2. For any point in the cell, this solution ensures that the corresponding sink has the minimum distance to the point. Each cell is regarded as a cluster and the partitions residing in the cell is assigned to the corresponding cluster. If nodes of a partition resides in multiple clusters, we identify the cluster to assign based on the location of the partition's centroid location. Note that in this approach there is a possibility of having clusters without any partitions. In that case, an MDC will not be required and it can be used by other clusters. Extra MDCs are made available to the clusters with longer tour lengths. It should be noted that multiple MDCs in the same cluster do not cause rendezvous waiting delay since they would be placed apart from each other equally and they can initiate movement at the same time.

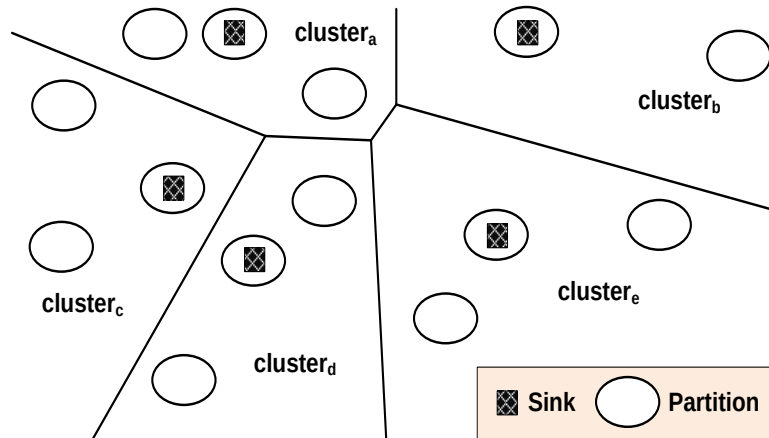


Figure 8.2: Clustering of partitions using Voronoi cells

Once the partitions are clustered, an MDC is assigned for each cluster to tour the partitions so that the dependency of MDCs as well as the waiting delay is avoided. A

sample illustration of the MDC tours in the clusters is represented in Fig. 8.3.

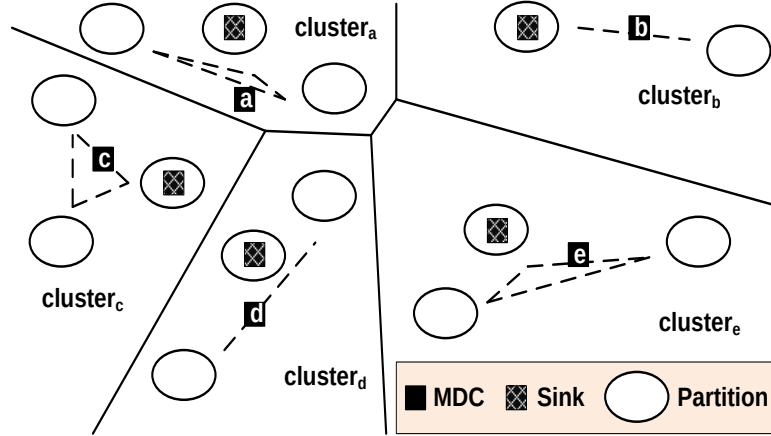


Figure 8.3: Forming tours in the clusters

8.4 EXPERIMENTAL EVALUATION

In order to evaluate the approaches, network topologies are created with disjoint partitions. A Java-based simulator is used to form the topologies and run the simulations. To deploy the nodes, an application area of $800 \times 800\text{m}$ is considered. Transmission range of the nodes is set to 50m. The approaches are tested under a varying number of sensors nodes, number of partitions and number of MDCs. For significance, 30 different topologies are created for each test case and the average is reported.

8.4.1 Performance Metric and Baselines

We consider average end-to-end delay of all the nodes as the main performance metric in this chapter. End-to-end delay is regarded as the time required for a packet to be transmitted across a network from source to destination. Minimizing this time is crucial for applications in general but specifically for real-time applications. This delay includes the transmission delay, touring delay and waiting delays as mentioned before. A constant value of 0.1ms is used as the transmission delay (including the propagation delay) between two hops. For touring delay, we assume a delay of 1 second for each 10 meter of movement.

For the single sink case, we considered the approach without any scheduling as one of

the baselines. This one, shown as *No Scheduling (NS)* in the graphs, is used to assess the impact of scheduling. Our proposed approach is shown as *Rendezvous Aware Scheduling (RAS)* in the graphs. As the baseline approach, we compare the delay to that of [36] when there is a single sink and k MDCs. This approach exploits k -TSP problem and strives to minimize the tour lengths. We implemented this approach with and without scheduling. The version with the scheduling is shown as *IDM- k -MDC-RAS* while the one without the scheduling is *IDM- k -MDC-NS*.

For the multiple sink case, our approach is represented as *Voronoi-based Clustering-Scheduled (VBCS)*. The approach was run under different number of sinks and compared to the best case with the single sink nodes which is *RAS*.

8.4.2 Experiment Results and Analysis

We conducted experiments to evaluate the approaches in terms of end-to-end delay. The results are presented for solutions assuming single sink and multiple sinks separately.

8.4.2.1 Experiment Results with Single Sink

In this subsection, we provide the experiment results of the solutions assuming single sink in the network. The results are depicted in Fig. 8.4, Fig. 8.5, and Fig. 8.6 for varying number of partitions, nodes, and MDC count respectively.

Fig. 8.4 portrays the results for varying number of partitions when the number of nodes and available MDC count is fixed. It can be observed from Fig. 8.4 that delay is increasing for all of the approaches when the number of partitions is increased. This can be explained as follows: For a partitioned network with a single sink, only the sensors within the same partition with the sink can deliver their data without using an MDC. The sensors in other partitions require an MDC to collect their data. For a constant number of sensors, if the number of partitions is increased, the number of sensors in the sink partition will be reduced and thus more sensors will require an MDC to send their data. As a result,

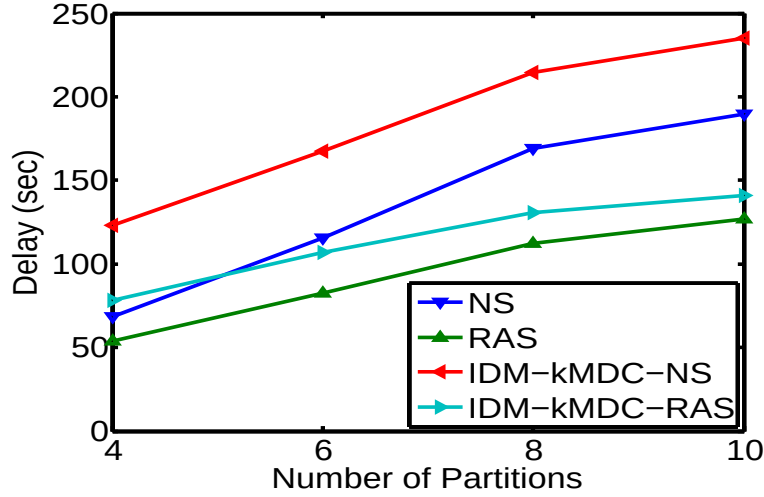


Figure 8.4: End-to-end delay of sensors for varying partition count. Node count: 100, MDC count: 2.

not only the tour lengths and MDC touring delays but also the rendezvous waiting delay increases.

It can also be observed from Fig. 8.4 that the proposed *RAS* significantly outperform *NS* and *IDM - kMDC* for both versions. This is due to the application of the scheduling algorithm in *RAS* and allowing additional tours when long waiting times is expected. Both *RAS* and *NS* outperform the respective versions of *IDM - kMDC*. This can be attributed to the underlying clustering algorithm used. Our clustering algorithm is based on [60] and provides a better load balancing than *IDM - kMDC* and thus this reduces the rendezvous waiting times.

Fig. 8.5 presents the results for varying number of nodes when the number of partitions and the available MDC count is fixed. It can be observed from Fig. 8.5 that delay is increasing when scheduling is not applied. When applied, the delay increases up to 150 nodes and then starts declining. Even though we provide the average delay per node, the increase in delay when the number of nodes is increased can be attributed to longer paths to reach the sink. However, upon reaching the threshold as in the application of scheduling, the delay starts to decline. This is due to the shorter distance between the partitions with the increased node density which causes shorter tour lengths. Our approach outperforms

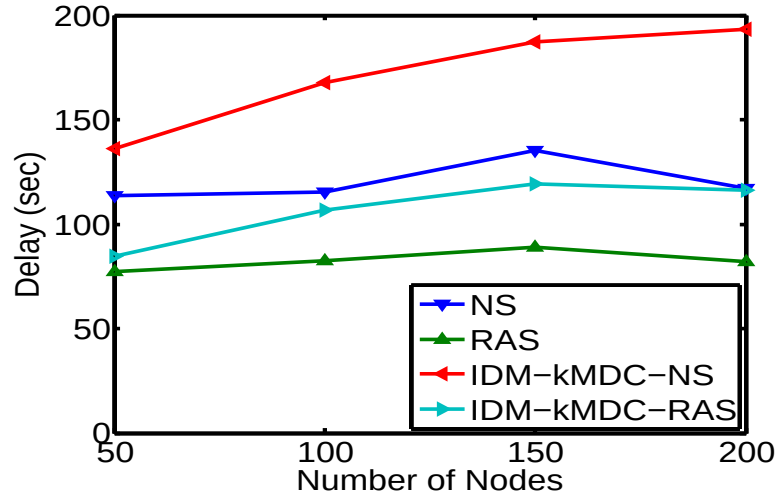


Figure 8.5: End-to-end delay of sensors for varying node count. Partition count:6, MDC count: 2.

IDM – kMDC – RAS again due to same reasons explained above.

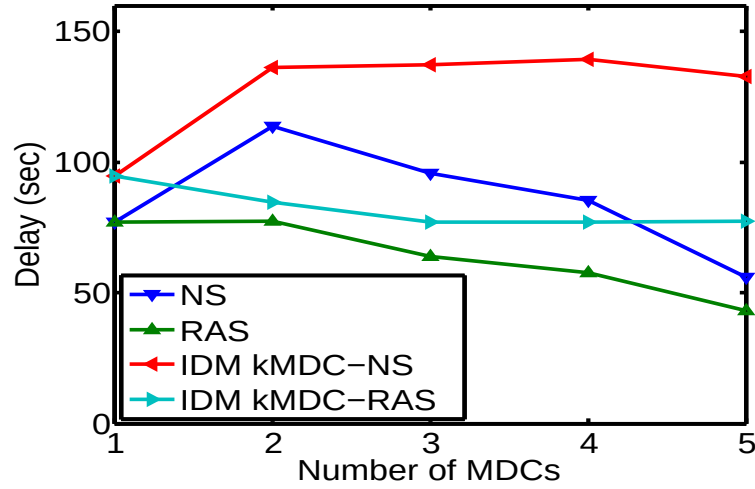


Figure 8.6: End-to-end delay of sensors for varying MDC count. Node count: 50, Partition count: 6.

Fig. 8.6 portrays the results for varying MDC count when the number of nodes and partition count is fixed. It can be observed from Fig. 8.6 that delay declines when the number of available MDCs is increased and scheduling is applied. On the other hand, delay increases initially and then declines when scheduling is not applied. Albeit the initial increase/decrease, the change in delay of *IDM – kMDC* for both versions is not very significant afterwards. Additional MDCs provide shorter tour lengths and less delay for

our approaches. However, due to the poor clustering and not applying the scheduling, waiting delay of more MDCs can increase in $IDM - kMDC$ which affects the overall end-to-end delay adversely.

8.4.2.2 Experiment Results with Multiple Sinks

In this subsection, we provide the experiment results of the proposed approach assuming multiple sinks in the network. The results are depicted in Fig. 8.7, Fig. 8.8, and Fig. 8.9 for varying number of partitions, nodes, and MDC count respectively.

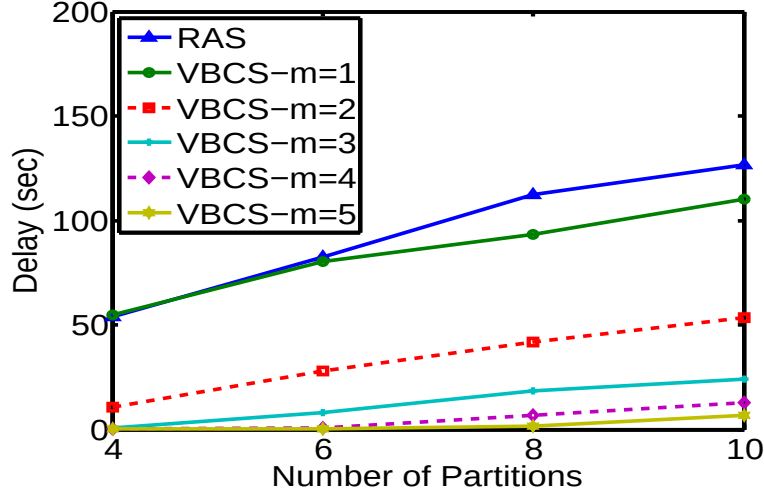


Figure 8.7: End-to-end delay of sensors for varying partition count. Node count: 100, MDC count for *RAS*: 2.

Fig. 8.7 portrays the results for varying number of partitions when the number of nodes is fixed. It can be observed from Fig. 8.7 that delay increases when the number of partitions is increased due to the reasons explained earlier. We also observe that delay reduces with the increased MDC count for *VBCS*. *VBCS* with a single MDC outperforms *RAS* due to a better clustering based on a good sink location and Voronoi cells. Recall that for *RAS*, the sink location is picked randomly.

Fig. 8.8 presents the results for varying number of nodes when the number of partitions is fixed. It can be observed from Fig. 8.8 that delay is increasing slightly for the *VBCS* with MDC count 1 when the number of nodes is increased. On the other hand, delay is

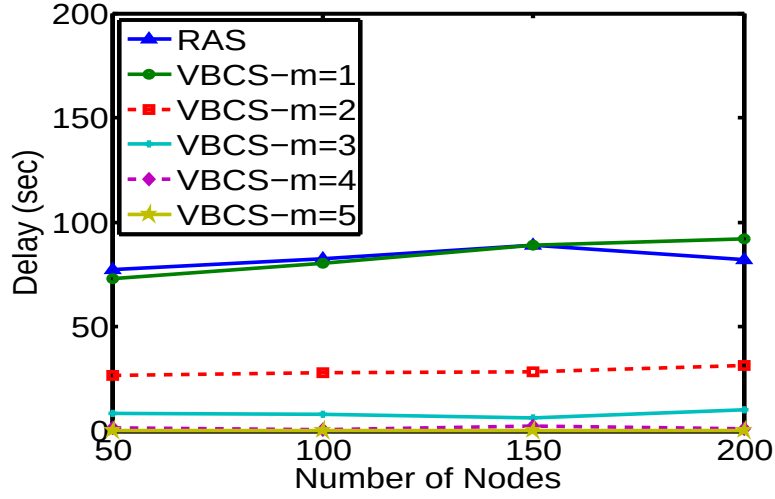


Figure 8.8: End-to-end delay of sensors for varying node count. Partition count: 6, MDC count for *RAS*: 2.

almost constant with multiple MDCs when the proposed approach is applied in networks with different node densities. The slight increase in delay can be attributed to the longer paths to reach the sink. Since the partition count is constant, delay is not affected by the node density significantly when multiple MDCs are available. *RAS* performs slightly better than the *VBCS* with 1 MDC. However, with 2 or more MDCs *VBCS* is still much better.

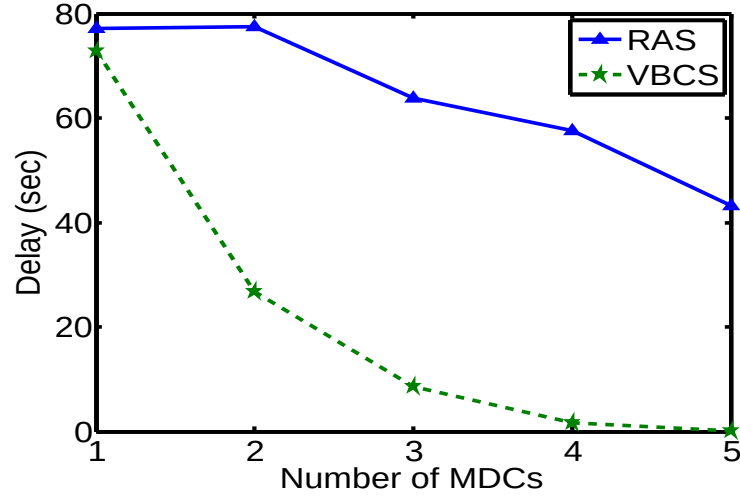


Figure 8.9: End-to-end delay of sensors for varying MDC count. Node count: 50, partition count: 6.

Fig. 8.9 portrays the results for varying MDC count when the number of nodes and partition count is fixed. It can be observed from Fig. 8.9 that delay declines for both

approaches when the number of available MDCs is increased. The decline in delay for *RAS* can be attributed to the decrease of the touring delays by the availability of MDCs in higher count. On the other hand, for *VBCS*, the number of clusters increases with the increased MDC count. Increasing the cluster count decreases the average distance to sink which helps to minimize the touring delays. Another advantage is avoiding rendezvous waiting delay which is avoided due to delivery of data to different sinks.

8.5 SUMMARY

In this chapter, we tackled the problem of rendezvous waiting in disjoint WSNs where connectivity is provided via MDCs in an intermittent manner. We first demonstrated that with the existing approaches where multiple MDCs are used to tour the partitions in disjoint WSNs, there will be situations where MDCs need to wait for other MDCs to receive their data and forward it to the sink. We proposed a scheduling algorithm to alleviate the problem based on the tour lengths of neighboring clusters' MDCs. However, since this does not completely eliminate the waiting problem, we proposed another approach with the use of multiple sinks. This approach aimed at parallelizing the data collection within each cluster and thus waiting is not experienced by the MDCs. We propose mechanism to identify the right sink locations and cluster the partitions around these locations such that every sink serves an MDC. The determination of sink locations was based on p -center optimization and the clustering followed the idea of Voronoi cells.

The evaluation of the proposed approaches revealed that rendezvous scheduling reduces the waiting delay significantly and thus our approach outperforms the existing approaches where rendezvous waiting is not considered. The experimentation with multiple sinks has also shown the significant impact of eliminating rendezvous waiting and minimizing the total delay.

CHAPTER 9

CONCLUSION AND FUTURE WORK

In this chapter, we summarize the completed research and discuss the future work.

9.1 COMPARISON OF THE PROPOSED APPROACHES

We present approaches to deal with the connectivity restoration in partitioned networks in this dissertation. We proposed approaches exploiting mobility of the existing nodes assuming introducing external nodes (i.e., relays) to the network is not always feasible nor possible. We have presented 6 novel approaches including a near-optimal solution:

- (a) *A Near Optimal Solution for Connectivity Restoration Using Existing Nodes:* This approach provides a near-optimal solution which can be used as a benchmark for the centralized approaches striving to recover connectivity of the partitioned networks. Even though providing a near-optimal solution, it is not scalable and computationally complex.
- (b) *An Effective and Scalable Connectivity Restoration Using Existing Nodes:* The idea of this approach is similar to the near-optimal solution however, instead of considering the infinite set of possible positions, only a limited set is considered to determine the destinations of the movements. This approach provides a better scalability comparing with the near-optimal solution and has performance close to the near-optimal solution.
- (c) *A Distributed Game Theoretic Approach for Connectivity Restoration Using Existing Nodes:* This approach does not assume a complete information about the topology and failures and guarantees connectivity with limited information in a distributed manner using local information.
- (d) *A Distributed Relay Placement Approach for Connectivity Restoration:* This approach

assumes the lack of complete information regarding the post-failure network state and guarantees connectivity through relay node placement in a distributed manner.

- (e) *A Relay Placement Approach for Connectivity Restoration using Limited Number of Relays:* This approach assumes unavailability of sufficient relay nodes to ensure connectivity for the whole network and assures overall network recovery by employing some of the relay nodes as mobile data collectors.
- (f) *A Quality-of-Service Aware Solution for Mobile Data Collector Assignment and Scheduling:* This approach considers the rendezvous waiting delay for mobile data collectors due to the inter-dependency of the MDCs and presents a solution to eliminate the rendezvous waiting delay completely.

Table 9.1 compares the features of the proposed approaches.

Table 9.1: Comparison of the Proposed Approaches

Approach	Centralized/Distributed	Scalable	Optimal
a	Centralized	No	Yes
b	Centralized	Yes	No
c	Distributed	Yes	No
d	Distributed	Yes	No
e	Centralized	Yes	No
f	Centralized	Yes	No

9.2 FUTURE WORK

We plan to continue conducting research on the problem of federating disjoint networks in the future by focusing on different aspects of the problem. We have considered employing relay nodes as mobile data collectors to tackle the problem of connectivity restoration and also presented a QoS-aware solution. We proposed a clustering approach exploiting voronoi cells to provide a fair distribution of the nodes in the clusters. As a future work, we are planning to improve the clustering algorithm and compare it with similar clustering approaches in the literature. Several new factors can be considered to improve clustering regarding the determined tours at the end for the mobile data collectors. Besides the velocity of the mobile data collectors, data exchange procedure can affect the data delay

significantly.

Current approaches regards only the distances to be travelled by the mobile data collectors while considering the delay and the partitions to be clustered. However, the number of stops as well as the number of nodes in the partitions should also be considered while clustering the partitions. The number of stops of a mobile data collector represents the data collection/exchange points on its path. The data collector is expected to stop at each point to initiate the neighbor discovery protocol through hello messages, establish connection with the representative node and complete data transfer. Delay is affected from the number of stops since the movement is interrupted at each stop and the time to reach the maximum speed may vary. The number of nodes in the partitions is another factor with a considerable impact on delay. The amount of data collected in a partition depends on the number of nodes in the partition. The higher amount of data collected in the partition extends the duration of data transfer between the data collector and the representative node. Thus, the number of nodes should also be considered while clustering. A possible approach to provide a balanced clustering while considering the number of stops and the number of nodes is using delay instead of tour lengths as a metric. Therefore, our solution will regard the total delay to be occurred including movement, delay at each stop and the data exchange delay due to the possible tour to be determined while clustering instead of the tour lengths only.s

REFERENCES

- [1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: A survey, Elsevier Computer Networks 38 (2002) 393–422.
- [2] G. T. Sibley, M. H. A. Rahimi, Robomote: A tiny mobile robot platform for large-scale ad-hoc sensor networks, in: IEEE International Conference on Robotics and Automation, 2002.
- [3] M. B. McMickell, B. Goodwine, L. A. Montestruque, Micabot: a robotic platform for large-scale distributed robotics, in: ICRA, 2003, pp. 1600–1605.
- [4] I. F. Akyildiz, I. H. Kasimoglu, Wireless sensor and actor networks: Research challenges, in: Elsevier Ad hoc Network Journal, Vol. 2, 2004, pp. 351–367.
- [5] G. Wang, G. Cao, T. L. Porta, W. Zhang, Sensor relocation in mobile sensor networks, in: Proceedings of the 24th International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM05), Miami, FL, March 2005.
- [6] A. Cerpa, D. Estrin, Ascent: Adaptive self-configuring sensor network topologies, SIGCOMM Comput. Commun. Rev. 32 (1) (2002) 62–62.
- [7] B. Chen, K. Jamieson, H. Balakrishnan, R. Morris, Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks, in: ACM Wireless Networks Journal, 2001, pp. 85–96.
- [8] Y. Xu, J. Heidemann, D. Estrin, Geography-informed energy conservation for ad hoc routing, in: MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking, ACM, New York, NY, USA, 2001, pp. 70–84.
- [9] L. Boloni, D. Turgut, Protecting bridges: Reorganizing sensor networks after catastrophic events, in: Federated Wireless Sensor Systems in conjunction with Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International, 2011, pp. 2028 –2033.
- [10] W. Zhang, G. Xue, S. Misra, Fault-tolerant relay node placement in wireless sen-

- sensor networks: Problems and algorithms, in: INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE, 2007, pp. 1649–1657. doi:10.1109/INFCOM.2007.193.
- [11] S. Lee, M. Younis, Recovery from multiple simultaneous failures in wireless sensor networks using minimum steiner tree, *J. Parallel Distrib. Comput.* 70 (2010) 525–536. doi:http://dx.doi.org/10.1016/j.jpdc.2009.12.004.
URL <http://dx.doi.org/10.1016/j.jpdc.2009.12.004>
- [12] F. Senel, M. Younis, Relay node placement in structurally damaged wireless sensor networks via triangular steiner tree approximation, *Computer Communications* 34 (16) (2011) 1932 – 1941. doi:10.1016/j.comcom.2011.05.010.
- [13] Q. Wang, G. Takahara, H. Hassanein, K. Xu, On relay node placement and locally optimal traffic allocation in heterogeneous wireless sensor networks, in: *Proc. of IEEE Local Computer Net. (LCN05)*, IEEE, 2005.
- [14] Q. Wang, G. Takahara, H. Hassanein, K. Xu, Locally optimal relay node placement in heterogeneous wireless sensor networks, in: *Proc. of IEEE GLOBECOM (GLOBECOM05)*, IEEE, 2005.
- [15] F. M. Al-Turjman, H. S. Hassanein, W. M. Alsalih, M. Ibnkahla, Optimized relay placement for wireless sensor networks federation in environmental applications, *Wireless Communications and Mobile Computing* 11 (12) (2011) 1677–1688.
- [16] M. Sir, I. Senturk, E. Sisikoglu, K. Akkaya, An optimization-based approach for connecting partitioned mobile sensor/actuator networks, in: *Proceedings of 3rd International Workshop on Wireless Sensor, Actuator and Robot Networks (WiSARN)*, in conjunction with IEEE INFOCOM'11), April 2011, Shanghai, China.
- [17] A. Abbasi, K. Akkaya, M. Younis, A distributed connectivity restoration algorithm in wireless sensor and actor networks, in: *Proceedings of IEEE International Conference on Local Computer Networks, (LCN'07)*, Dublin, Ireland, Oct. 2007.
- [18] K. Akkaya, F. Senel, A. Thimmapuram, S. Uludag, Distributed recovery of actor

- failures in wireless sensor and actor networks, in: Proceedings of IEEE Wireless Communications and Networking Conference (WCNC'08), Las Vegas, NV, Sept. 2008.
- [19] P. Basu, J. Redi, Movement control algorithms for realization of fault-tolerant ad hoc robot networks, in: IEEE Networks Journal, Vol. 18, Aug. 2004, pp. 36–44.
 - [20] A. Abbasi, M. Younis, K. Akkaya, Movement-assisted connectivity restoration in wireless sensor and actor networks, in: IEEE Transactions on Parallel and Distributed Systems, Vol. 20, Sept. 2009, pp. 1366–1379.
 - [21] K. Akkaya, F. Senel, A. Thimmapuram, S. Uludag, Distributed recovery from network partitioning in movable sensoractor networks via controlled, in: IEEE Transactions on Computers, Vol. 59, Feb. 2010, pp. 258–271.
 - [22] A. Abbasi, M. Younis, U. Baroudi, Restoring connectivity in wireless sensor-actor networks with minimal topology changes, in: 2010 IEEE International Conference on Communications (ICC), Cape Town, South Africa, May 2010.
 - [23] K. Akkaya, F. Senel, A. Thimmapuram, S. Uludag, Distributed recovery from network partitioning in movable sensor/actor networks via controlled mobility, Computers, IEEE Transactions on 59 (2) (2010) 258 –271.
 - [24] H. Liu, X. Chu, Y. Leung, R. Du, Simple movement control algorithm for bi-connectivity in robotic sensor networks, IEEE Journal on Selected Areas in Communications 28 (7) (2010) 994 – 1005.
 - [25] F. Senel, K. Akkaya, M. Younis, An efficient mechanism for establishing connectivity in wireless sensor and actor networks, in: IEEE Global Telecommunications Conference (GLOBECOM), 2007, Washington, DC, Nov. 2007.
 - [26] A. Alfadhly, U. Baroudi, M. Younis, Optimal node repositioning for tolerating node failure in wireless sensor actor network, in: Proceedings of 25th Biennial Symposium on Communications (QBSC), Ontario, Canada, May 2010.
 - [27] W. Wang, V. Srinivasan, K.-C. Chu, Using mobile relays to prolong the lifetime of wireless sensor networks, in: Proceedings of the 11th Annual international conference

- on Mobile computing and networking (Mobicom05), Cologne, Germany, Aug. 2005.
- [28] E. Lloyd, G. Xue, Relay node placement in wireless sensor networks, in: IEEE Trans. on Computers, Vol. 56, 2007, pp. 134–138.
 - [29] X. Han, X. Cao, E. Lloyd, C.-C. Shen, Fault-tolerant relay node placement in heterogeneous wireless sensor networks, in: INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE, 2007, pp. 1667–1675. doi:10.1109/INFCOM.2007.195.
 - [30] J. Tang, B. Hao, A. Sen, Relay node placement in large scale wireless sensor networks, Computer Communications 29 (4) (2006) 490 – 501, current areas of interest in wireless sensor networks designs. doi:DOI: 10.1016/j.comcom.2004.12.032.
 - [31] F. Senel, M. Younis, Optimized interconnection of disjoint wireless sensor network segments using k mobile data collectors, in: 2012 IEEE International Conference on Communications (ICC), Ottawa, Canada, June 2012.
 - [32] J. L. Stanislaus, M. Younis, Delay-conscious federation of multiple wireless sensor network segments using mobile relays, in: 76th IEEE Vehicular Technology Conference (VTC2012-Fall), Quebec City, Canada, September 2012.
 - [33] H. M. Almasaeid, A. E. Kamal, Data delivery in fragmented wireless sensor networks using mobile agents, in: Proceedings of the 10th ACM Symposium on Modeling, analysis, and simulation of wireless and mobile systems, MSWiM '07, ACM, New York, NY, USA, 2007, pp. 86–94. doi:10.1145/1298126.1298143.
URL <http://doi.acm.org/10.1145/1298126.1298143>
 - [34] H. M. Almasaeid, A. E. Kamal, Modeling mobility-assisted data collection in wireless sensor networks., in: GLOBECOM, IEEE, 2008, pp. 188–192.
 - [35] W. Zhao, M. H. Ammar, E. W. Zegura, A message ferrying approach for data delivery in sparse mobile ad hoc networks., in: J. Murai, C. E. Perkins, L. Tassiulas (Eds.), MobiHoc, ACM, 2004, pp. 187–198.
URL <http://dblp.uni-trier.de/db/conf/mobihoc/mobihoc2004.html#ZhaoAZ04>

- [36] F. Senel, M. Younis, Optimized interconnection of disjoint wireless sensor network segments using k mobile data collectors, in: IEEE International Conference on Communications (ICC'12), Ottawa, Canada, June 2012.
- [37] H. Almasaeid, A. E. Kamal, Data delivery in fragmented wireless sensor networks using mobile agents, in: 10th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), Chania, Greece, October 2007.
- [38] J. L. Stanislaus, M. Younis, Delay-conscious federation of multiple wireless sensor network segments using mobile relays, in: IEEE Vehicular Technology Conference (VTC2012-Fall), Qubec City, Canada, September 2012.
- [39] A. Abbas, M. Younis, Interconnecting disjoint network segments using a mix of stationary and mobile nodes, in: IEEE Conference on Local Computer Networks (LCN 2012), Clearwater, FL, to appear.
- [40] L. Hu, D. Evans, Localization for mobile sensor networks, in: Proceedings of International Conference on Mobile Computing and Networking (MobiCom'04), Philadelphia, Pennsylvania, Sept. 2004.
- [41] G.-H. Lin, G. Xue, Steiner tree problem with minimum number of steiner points and bounded edge-length, *Inf. Process. Lett.* 69 (1999) 53–57.
- [42] P. K. Nagilla, Sensor Coverage and Actors Relocation in Wireless Sensor and Actor Networks (WSAN): Optimization Models and Approximation Algorithms, Master's thesis, University of Missouri, Columbia, Missouri (2010).
- [43] F. Senel, M. Younis, Optimized connectivity restoration in a partitioned wireless sensor network, in: Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE, 2011, pp. 1 –5. doi:10.1109/GLOCOM.2011.6134397.
- [44] F. Dai, J. Wu, An extended localized algorithms for connected dominating set formation in ad hoc wireless networks, in: IEEE Trans. on Parallel and Distributed Systems, Vol. 15, 2004.

- [45] A. Rowe, C. Rosenberg, I. Nourbakhsh, A low cost embedded color vision system, in: Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on, Vol. 1, 2002, pp. 208 – 213 vol.1. doi:10.1109/IRDS.2002.1041390.
- [46] X. Li, H. Shi, Y. Shang, Sensor network localisation based on sorted rssi quantisation, International Journal of Ad Hoc and Ubiquitous Computing 1 (4) (2010) 222–229.
- [47] Depth of Field, http://en.wikipedia.org/wiki/Depth_of_field.
- [48] A. S. Galeotti, M. O. Goyal, F. Jackson, L. Y. Vega-Redondo, Network games, in: Review of Economic Studies, Vol. 77, 2010, pp. 218–244.
- [49] S. Vemulapalli, K. Akkaya, Mobility-based self route recovery from multiple node failures in mobile sensor networks, in: Proceedings of 10th IEEE International Workshop on Wireless Local Networks (WLN) in conjunction with LCN’10, Denver, CO, Oct. 2010.
- [50] I. Downes, L. B. Rad, H. Aghajan, Development of a mote for wireless image sensor networks, In proc. Cognitive Systems with Onteractive Sensors.
- [51] A. Rowe, A. Goode, D. Goel, I. Nourbakhsh, Cmucam3: an open programmable embedded vision sensor, Technical Report, RI-TR-07-13, Carnegie Mellon Robotics Institute.
- [52] Pioneer P3-DX Robot, <http://www.mobilerobots.com/>.
- [53] S. Soro, W. Heinzelman, A Survey of Visual Sensor Networks, Advances in Multimedia 2009 (2009) 1–22. doi:10.1155/2009/640386.
URL <http://dx.doi.org/10.1155/2009/640386>
- [54] H. Oztarak, K. Akkaya, A. Yazici, Lightweight object localization with a single camera in wireless multimedia sensor networks, in: IEEE Global Telecommunications Conference (GLOBECOM’09), Honolulu, Hawaii, Dec. 2009.
- [55] K. Akkaya, M. Younis, Coverage-aware and connectivity-constrained actor positioning in wireless sensor and actor networks, in: Proceedings of the 26th IEEE International Performance Computing and Communications Conference (IPCCC 2007), April 2007,

New Orleans, Louisiana.

- [56] G. Tan, S. Jarvis, A.-M. Kermarrec, Connectivity-guaranteed and obstacle-adaptive deployment schemes for mobile sensor networks, *Mobile Computing, IEEE Transactions on* 8 (6) (2009) 836–848. doi:10.1109/TMC.2009.31.
- [57] K. Akkaya, S. Janapala, Maximizing connected coverage via controlled actor relocation in wireless sensor and actor networks, in: *Elsevier Computer Networks*, Vol. 52, 2008, pp. 2779–2796.
- [58] K. A. I.F. Senturk, F. Senel, An effective and scalable connectivity restoration heuristic for mobile sensor/actor networks, in: *IEEE Global Telecommunications Conference (GLOBECOM'12)*, 2012.
- [59] D. L. Applegate, R. E. Bixby, V. Chvatal, W. J. Cook, *The Traveling Salesman Problem: A Computational Study* (Princeton Series in Applied Mathematics), Princeton University Press, Princeton, NJ, USA, 2007.
- [60] I. Senturk, K. Akkaya, F. Senel, M. Younis, Connectivity restoration in disjoint wireless sensor networks using limited number of mobile relays, in: *Communications (ICC), 2013 IEEE International Conference on*, (to appear).
- [61] N. Mladenović, M. Labbé, P. Hansen, Solving the p-center problem with tabu search and variable neighborhood search, *Networks* 42 (1) (2003) 48–64.
- [62] B. A. Bash, P. Desnoyers, Exact distributed voronoi cell computation in sensor networks, in: T. F. Abdelzaher, L. J. Guibas, M. Welsh (Eds.), *Proceedings of the 6th International Conference on Information Processing in Sensor Networks, IPSN 2007*, Cambridge, Massachusetts, USA, April 25-27, 2007, ACM, 2007, pp. 236–243. doi:http://doi.acm.org/10.1145/1236360.1236393.

VITA

Graduate School
Southern Illinois University

Izzet Fatih Senturk

Date of Birth: November 21, 1984

2015 Evergreen Terrace, Carebondale, Illinois 62901

ifs5@cornell.edu

Cornell University
Master of Engineering, Computer Science, May 2008

Ege University
Bachelor of Science, Computer Engineering, July 2006

Special Honors and Awards:

- Student Grant, 37th Annual IEEE Conference on Local Computer Networks (LCN'12), October 22-25, Clearwater, Florida, USA
- Student Travel Grant, International Conference on Communications (ICC'12), June 10-15, 2012, Ottawa, Canada
- Graduate Research Fellowship Honorable Mention, Ministry of Education - TURKEY, 2007-2008

Research Paper Title:

Restoring Connectivity in Partitioned Wireless Sensor Networks

Major Professor: Dr. K. Akkaya

Publications:

Book Chapters:

- Akkaya K., **Senturk I.F.**, and Janansefat S., Design, Implementation and Networking of Mobile Sensing Platforms, *Cooperative Robots and Sensor Networks (second edition)*, Springer (submitted)
- Younis M., Lee S., **Senturk I.F.**, and Akkaya K., Topology Management Techniques for Tolerating Node Failure in WSNs, *The Art of Wireless Sensor Networks*, Ed. Habib Ammari, Springer (to appear)

Journals:

- **Senturk I.F.**, Akkaya K., and Janansefat S., Terrain aware Connectivity Restoration in Partitioned Mobile Sensor Networks due to Single and Multiple Node Failures (submitted)
- **Senturk I.F.**, Akkaya K., and Yilmaz S., Distributed Relay Node Placement under Limited Information for Restoring Connectivity in Partitioned Wireless Sensor Networks, *Ad Hoc Networks Journal* (to appear)
- Younis M., **Senturk I.F.**, Akkaya K., Lee S., and Senel F., Topology Management Techniques for Tolerating Node Failures in Wireless Sensor Networks: A Survey, *The International Journal of Computer and Telecommunications Networking* (to appear)
- **Senturk I.F.**, Yilmaz S., and Akkaya K., Connectivity Restoration in Delay Tolerant Sensor Networks using Game Theory, *International Journal of Ad Hoc and Ubiquitous Computing* (Available Online)
- Akkaya, K., **Senturk, I.F.**, and Vemulapalli S., Handling Large-scale Node Failures in Mobile Sensor/Robot Networks, *Journal of Network and Computer Applications* (Available Online)

Conferences:

- **Senturk I.F.** and Akkaya K., Mobile Data Collector Assignment and Scheduling for Minimizing Data Delay in Partitioned Wireless Sensor Networks (submitted)
- **Senturk I.F.** and Akkaya K., Energy and Coverage Trade-offs in Deploying a Mix of Mobile and Stationary Relays for Disjoint Wireless Sensor Networks, *IEEE Global Communications Conference 2013 (GLOBECOM'13)*, Dec. 9-13, 2013, Atlanta, GA, USA (to appear)
- Janansefat S., Akkaya K., **Senturk I.F.** and Gloff M., Rethinking Connectivity Restoration in WSNs Using Feedback from a Low-cost Mobile Sensor Network Testbed, *9th IEEE International Workshop on Performance and Management of Wireless and Mobile Networks (P2MNet'13)*, October 21-24, 2013, Sydney, AUSTRALIA (to appear)

- **Senturk I.F.**, Akkaya K., Senel F., Younis M., Connectivity Restoration in Disjoint Wireless Sensor Networks using Limited Number of Mobile Relays, *International Conference on Communications (ICC'13)*, June 9-13, 2013, Budapest, HUNGARY
- **Senturk I.F.**, Akkaya K., and Senel F., An Effective and Scalable Connectivity Restoration Heuristic for Mobile Sensor/Actor Networks, *IEEE Global Communications Conference 2012 (GLOBECOM'12)*, Dec. 3-7, 2012, Anaheim, CA, USA
- **Senturk I.F.** and Akkaya K., Energy and Terrain Aware Connectivity Restoration in Disjoint Mobile Sensor Networks, *12th IEEE International Workshop on Wireless Local Networks (WLN'12)*, Oct. 22-25, 2012, Clearwater, FL, USA
- **Senturk I.F.** and Akkaya K., On the Performance of Sensor Node Repositioning Under Realistic Terrain Constraints (short paper), *37th Annual IEEE Conference on Local Computer Networks (LCN'12)*, Oct. 22-25, 2012, Clearwater, FL, USA
- **Senturk I.F.**, Yilmaz S., and Akkaya K., Distributed Relay Node Positioning for Connectivity Restoration in Partitioned Wireless Sensor Networks, *IEEE Symposium on Computers and Communication (ISCC'12)*, July 1-4, 2012, Cappadocia, TURKEY
- **Senturk I.F.**, Yilmaz S., and Akkaya K., A Game-Theoretic Approach to Connectivity Restoration in Wireless Sensor and Actor Networks, *International Conference on Communications (ICC'12) - WiSAAN*, June 10-15, 2012, Ottawa, CANADA
- Sir M., **Senturk I.F.**, Sisikoglu E., and Akkaya K., An Optimal Algorithm For Connecting Partitioned Mobile Sensor/Actuator Networks, *The Third International Workshop on Wireless Sensor, Actuator and Robot Networks*, April 10-15, 2011, CHINA