



**Movement-Assisted and Application-Aware
Connectivity Restoration in
Wireless Sensor-Actor Networks**

BY

Ameer Ahmed Abbasi

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER ENGINEERING

June 2011

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN 31261, SAUDI ARABIA


DEANSHIP OF GRADUATE STUDIES

This thesis, written by **AMEER AHMED ABBASI** under the direction of his thesis advisor and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE in COMPUTER ENGINEERING**.

Thesis Committee



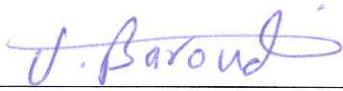
Department Chairman
Dr. Basem Al-Madani




Dean of Graduate Studies
Dr. Salam A. Zummo




2/7/11
Date



Thesis Advisor
Dr. Uthman A. Baroudi



Member
Dr. Mayez A. Al-Mouhamed



Member
Dr. Wasfi G. Al-Khatib

DEDICATIONS



Dedicated to

Dr. Mohamed Younis

&

My family Members

Without them it should never have been completed

ACKNOWLEDGMENT

All praises, all glory and all thanks are due to Allah, The Majestic, The Almighty for bestowing me with knowledge, guidance, patience, courage and health to achieve this work. May peace and blessings be upon prophet Mohammed (PBUH), his family and his companions.

I would like to acknowledge the King Fahd University of Petroleum & Minerals for the support extended towards my research and providing me the opportunity to pursue graduate studies.

I wish to express my sincere gratitude and appreciation to *Dr. Uthman A. Baroudi* who served as my thesis advisor, provided me substantial motivation, technical knowledge and philosophical guidance and supported me all the way relentlessly. I also wish to thank the other members of my thesis committee *Dr. Mayez A. Al-Mouhamed* and *Dr. Wasfi G. Al-Khatib* for their constructive support and encouragement.

I am really grateful to my source of inspiration, mentor and friend *Dr. Mohamed Younis* for his constant endeavor, guidance, positive criticism and the numerous moments of attention that he devoted throughout my research work. His valuable suggestions made the research work interesting and knowledgeable for me.

Very special thanks to my beloved mother and father. Both have been a source of encouragement and inspiration to me throughout my life. I am very grateful for the myriad of ways in which, throughout my life, they both have actively supported me in my determination to find and realize my potential.

A very special thanks to my dear wife, who remains willing to engage with the struggle, and ensuing discomfort, of having a partner who is busy with the studies, research, work and business even when she needs him on her side. I am really grateful for her continuous practical and emotional support to the competing demands of business, work, study and personal development. Love and thanks to dear Maria, Hala and Marwa for being so supportive and tolerant - even when being 'without father' was hard.

Finally, I thank to all my family members for their continuous love, encouragement, prayers, emotional and moral support throughout my life. Words fall short in conveying my gratitude towards them. A prayer is the simplest way I can repay them – May Allah (S.W.T) give them good health and give me ample opportunity to be of service to them throughout my life.

TABLE OF CONTENTS

DEDICATIONS	III
ACKNOWLEDGMENT	IV
TABLE OF CONTENTS	VI
LIST OF TABLES	X
LIST OF FIGURES.....	XI
THESIS ABSTRACT (ENGLISH).....	XIV
THESIS ABSTRACT (ARABIC).....	XV
CHAPTER 1.....	1
INTRODUCTION.....	1
1.1 CONNECTIVITY ISSUE FOR WIRELESS SENSOR-ACTOR NETWORKS	2
1.2 CONTRIBUTION OF THESIS	5
1.3 SYSTEM MODEL	6
1.4 PROBLEM FORMULATION & METHODOLOGY	9
1.4.1 Topology Repair With Application Level Constraints On Mobility	9
1.4.2 Least Disruptive Topology Repair.....	12
1.4.3 Least Movement Topology Repair	15
1.5 ORGANIZATION OF THESIS	17
CHAPTER 2.....	18
LITERATURE REVIEW	18
2.1 RECOVERY THROUGH NODE REPOSITIONING	19
2.2 RECOVERY BY PLACEMENT OF RELAY NODES	25
CHAPTER 3.....	27
EXPERIMENTAL SETUP & PERFORMANCE METRICS.....	27
3.1 INTRODUCTION	27
3.2 OVERVIEW OF WSAN SIMULATOR.....	27
3.2.1 General -Purpose	28

3.2.2	Design.....	28
3.2.3	Extensible	30
3.2.4	Graphical User Interface (GUI).....	30
3.3	NETWORK TOPOLOGIES & ENVIRONMENT	32
3.4	SIMULATION ASSUMPTIONS.....	34
3.5	PERFORMANCE METRICS.....	35
CHAPTER 4.....		37
CONNECTIVITY RESTORATION WITH APPLICATION MOBILITY CONSTRAINTS		37
4.1	INTRODUCTION	37
4.2	DETAILED C ² AM STEPS	38
4.2.1	Maintaining a List of 2-hop Neighbors	38
4.2.2	Detecting a Failure and Initiating the Recovery Process	39
4.2.3	Application-Aware Qualification for Movement Test.....	40
4.2.4	Cascaded Relocation & Algorithm Termination	41
4.3	APPLICATION-AWARE RECOVERY: EXAMPLES.....	42
4.4	C ² AM PSEUDO CODE	44
4.5	PERFORMANCE EVALUATION OF C ² AM	46
4.5.1	MRI Performance	48
4.5.2	Movement Performance	50
4.5.3	Communication overhead.....	50
4.6	CONCLUDING REMARKS ON C ² AM.....	51
CHAPTER 5.....		52
CONNECTIVITY RESTORATION WITH MINIMAL TOPOLOGY CHANGES		52
5.1	INTRODUCTION	52
5.2	MAJOR STEPS OF LEDiR ALGORITHM	54
5.2.1	Failure Detection	54
5.2.2	Smallest Block Identification	55
5.2.3	Replacing Faulty Node	56

5.2.4	Children Movement.....	57
5.3	EXAMPLE SCENARIOS OF LEDiR.....	59
5.4	DISTRIBUTED LEDiR IMPLEMENTATION.....	61
5.5	LEDiR PSEUDO CODE.....	64
5.6	ALGORITHM ANALYSIS.....	66
5.7	PERFORMANCE EVALUATION OF LEDiR.....	74
5.7.1	Overhead Related Metrics.....	76
5.7.2	Path Length Validation Metrics.....	81
5.7.3	General Comments.....	85
5.8	CONCLUDING REMARKS ON LEDiR.....	86
CHAPTER 6.....		87
RESTORING CONNECTIVITY WITH MINIMAL NODE MOVEMENT.....		87
6.1	INTRODUCTION.....	87
6.2	LEMOTOR – MAIN STEPS.....	88
6.2.1	Failure Detection.....	89
6.2.2	Smallest Block Identification.....	89
6.2.3	Replacing the Faulty Node.....	89
6.2.4	Children Movement.....	90
6.3	RECOVERY WITH MINIMAL NODE MOVEMENT: AN EXAMPLE.....	90
6.4	DISTRIBUTED LEMOTOR IMPLEMENTATION.....	92
6.5	LEMOTOR PSEUDO CODE.....	93
6.6	PERFORMANCE EVALUATION OF LEMOTOR.....	95
6.7	CONCLUDING REMARKS ON LEMOTOR.....	100
CHAPTER 7.....		101
CONCLUSION & FUTURE WORK.....		101
7.1	CONCLUSION.....	101
7.2	PUBLICATIONS.....	102
7.3	FUTURE WORK.....	104

APPENDIX A	105
REFERENCES	108
VITA	114

LIST OF TABLES

TABLE 2.1: Node relocation schemes that use metrics other than connectivity	19
TABLE 4.1: Attributes of the actors in Figure 1.2-(a)	43
TABLE 4.2: Total # of Messages Sent by C ² AM with varying # of actors.....	51
TABLE 5.1: The Path Predecessor Matrix generated by the Floyd-Warhsell algorithm [41] for the network topology of Figure 1.3-(a). For each pair of nodes v and w , the path matrix entry $P[v,w]$ contains a node k which is the direct predecessor of w on the shortest path to v	55
TABLE 5.2: # of Messages Sent by LeDiR with varying # of actors	79
TABLE 5.3: # of Messages Sent by LeDiR with varying actor radio range	79
TABLE 6.1: # of Messages Sent by LeMoToR with varying # of actors	99
TABLE 6.2: # of Messages Sent by LeMoToR with varying actor radio range	99
TABLE A.1: Statistical Analysis of C ² AM.....	106
TABLE A.2: Statistical Analysis of LeDiR	106
TABLE A.3: Statistical Analysis of LeMoToR	107

LIST OF FIGURES

Figure 1.1: An articulation of a WASN with a connected inter-actor network.	7
Figure 1.2: (a) Pre-failure network topology; (b) After A_1 fails the network gets partitioned into three disjointed sub-networks; (c) By using [10], node A_3 replaced the faulty actor and reestablished connectivity between actors; (d) The topology after running C ² AM with node A_5 replacing A_1 , followed with cascaded motion of A_6 , A_9 and A_{11}	11
Figure 1.3: Illustration on how DARA [10] restores connectivity (a) Initial 1-connected WSAN topology (b) Disjointed network with faulty node A_{10} and potential best candidates A_3 , A_9 , A_{11} , and A_{14} (c) Based on least node degree, node A_{11} has been selected as best candidate to replace the faulty node A_{10} (d) Repaired topology with the highlighted nodes A_{11} , A_{12} , A_2 and A_{13} that participated in the recovery process.	13
Figure 1.4: Illustrating how RIM [11] restores connectivity after the failure of node A_{10} in the connected inter-actor topology of Figure 1.3-(a). Highlighted nodes are moved and get involved in the recovery process.	16
Figure 2.1: In a), actor A_1 is a dominatee and cannot be a cut vertex. A_2 is a dominator and has a dominatee A_1 which is not connected. Thus, A_2 is a cut-vertex. A_3 is also a cut vertex in a) but will not be a cut-vertex in b) [12].	21
Figure 2.2: An example for how RIM restoration process; each shaded node moves based on the positions of its neighbors, denoted in double -lined circles [11].	23
Figure 3.1: High-level block diagram of WSAN simulator [42].	29
Figure 3.2: Inter-actor 1-connected network topology in WSAN simulator.	31
Figure 3.3: Network topology with faulty cut-vertex actor node. Topology is disjointed into 3 sub-networks.	33
Figure 4.1: Pseudo code for the C ² AM algorithm.	45

Figure 4.2: Measure of disturbance of application with varying actor count (Radio Range = 100m).....	47
Figure 4.3: Level of disturbance to the application under varying actor radio range (with 60 actors).....	47
Figure 4.4: Total distance traveled with varying number of actors (Radio Range = 100m).....	49
Figure 4.5: Travel distance with varying actor radio range (60 actors).....	49
Figure 5.1: Illustrating the movement of block B_s in LeDiR to restore the network connectivity and to keep intra-block paths unchanged; (a) that entire B_s moved r units (b) the collective effect of B_s participation in the recovery is stretching B_s towards F , and (c) B_s is both stretched and moved with links within the B_s stretched in order to minimize the total travel distance. r is the actor's communication range.....	58
Figure 5.2: An example illustrating how LeDiR restores connectivity after the failure of node A_{10}	60
Figure 5.3: Pseudo code for the LeDiR algorithm.....	65
Figure 5.4: LeDiR restores the network connectivity after the failure of a cut-vertex (critical node); (a) shows a WSAN before a cut-vertex fails and (b) shows the WSAN topology after applying LeDiR.....	67
Figure 5.5: The worst case scenario topology where $N = 7$ and failure of A_4 has partitioned the network into two $(N-1)/2$ nodes blocks. LeDiR would involve maximum $(N-1)/2$ actors in the recovery process either A_3 or A_5 selected to replace the faulty node followed by a series of inter-block node relocation.	69
Figure 5.6: Assuming the worst case scenario presented in Figure 5.5, LeDiR selected A_3 to replace the faulty node A_4 by traveling distance r . Once A_3 moved to the new position, A_2 will move behind it to maintain direct connectivity. Later, A_1 will do the same. Since the network is 1-dimensional and nodes are located r units away from each other, the maximum distance travel by a node is r	72
Figure 5.7: (a) Effect of the network size on the total distance traveled by actor nodes under RIM and LeDiR where CL is varying (with $r=100$); (b) The impact of an increased actor's communication range on the relocation overhead for a network of 100 actor nodes.....	75

Figure 5.8: Number of actors that moved during the recovery while varying (a) the communication range (with $N = 100$), and (b) the network size (with $r = 100$).	77
Figure 5.9: The number of extended paths per topology [average over 30 runs] after performing the recovery while varying (a) the communication range (with $N=100$), and (b) the network size (with $r=100$).	80
Figure 5.10: Percentage of shortest paths that are NOT extended per topology during the network recovery [average over 30 runs] while varying (a) the communication range (with $N=100$), and (b) the network size (with $r=100$).	82
Figure 5.11: (a) WSN with a sparse 1-connected topology and a faulty node. (b) Topology recovered by using RIM (c) topology recovered by using DARA and (d) topology recovered by using LeDiR	84
Figure 6.1: An example illustrating how LeMoToR restores connectivity after the failure of node A_{10}	91
Figure 6.2: Pseudo code of LeMoToR	94
Figure 6.3: The total distance traveled by actor nodes where (a) network size is varied (with $r=100$), (b) communication range is varied (with $N=100$)	96
Figure 6.4: Number of actors that moved during the recovery while varying (a) the network size (with $r = 100$), (b) the communication range (with $N = 100$).	98

THESIS ABSTRACT (ENGLISH)

NAME: Ameer Ahmed Abbasi
TITLE: Movement-Assisted and Application-Aware Connectivity Restoration in Wireless Sensor-Actor Networks
MAJOR: Computer Engineering
DATE: June 2011

In Wireless Sensor-Actor Networks (WSANs), sensors probe their surroundings and inform actors which respond collaboratively to achieve some desired application missions. Since actors have to coordinate their operation, it is necessary to maintain a strongly connected network topology at all time. A WSAN may get partitioned into disjoint segments, if a critical actor, i.e., a cut-vertex node, fails and causes the loss of multiple inter-actor communication links. The other actors close to the faulty node often exploit their mobility to autonomously restore the lost inter-actor connectivity. We propose C^2AM ; a recovery algorithm that factors in application level constraints on actor's mobility while restoring the network connectivity. In addition to considering physical level requirements, C^2AM accounts for application level concerns as well in order to avoid major disruptions to ongoing missions. Simulation results have validated the effectiveness of the proposed algorithm in maintaining both objectives. Moreover, we investigate the connectivity restoration problem subject to inter-actor communication path length constraints in order to handle the data latency and potential packet loss. We propose a Least-Disruptive topology Repair (LeDiR) algorithm. Unlike contemporary schemes that maintain 1 or 2-hop neighbor lists, LeDiR utilizes existing path discovery activities in the network in order to know the structure of the topology and avoids imposing additional pre-failure communication overhead. The performance of LeDiR is analyzed mathematically and validated via extensive simulation experiments. We extend LeDiR and name it Least-Movement Topology Repair (LeMoToR) algorithm. Like LeDiR scheme, LeMoToR is a distributed scheme that relies on the local view of a node about the network. However, LeMoToR does not impose any constraint to sustain the path length between any pair of node at pre-failure status and apply itself recursively to maintain network connectivity. To restore connectivity, LeMoToR strives to relocate the least number of nodes and reduce the traveled distance and message complexity. The simulation results validate the effectiveness of LeMoToR.

MASTER OF SCIENCE DEGREE
KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN, SAUDI ARABIA

THESIS ABSTRACT (ARABIC)

الاسم: أمير أحمد العباسي

عنوان الرسالة: استعادة الربط في شبكات الاستشعار اللاسلكية والأجهزة الفاعلة باستخدام تطبيقات التحريك والادراك

التخصص: هندسة الحاسب الآلي

تأريخ التخرج: حزيران 2011

في شبكات الاستشعار اللاسلكية والتشغيل الميكانيكية، (WSANs)، تقوم أجهزة الاستشعار بفحص محيطها وإبلاغ الأجهزة الفاعلة التي تستجيب بشكل تعاوني لتحقيق بعض بعثات التطبيق المطلوبة. بما أن من مهام الأجهزة الفاعلة تنسيق عملها، فمن الضروري الحفاظ على طوبولوجيا شبكة متصلة بقوة في جميع الأوقات. قد يتم تقسيم WSAN إلى قطاعات منفصلة، إذا كان الجهاز الفاعل الحساس، أي عقدة قطع الرأس، قد فشل وتسبب بفقدان العديد من وصلات الاتصال بين الأجهزة الفاعلة. كثيرا ما تستغل الأجهزة الفاعلة الأخرى القريبة من العقدة ذات العيوب عملها لاستعادة قدرتها على الحركة بشكل مستقل واسترجاع الاتصال المفقود بين الأجهزة الفاعلة. نقتح C^2AM ؛ وهي خوارزمية للإصلاح تقوم بالأخذ بعين الاعتبار القيود على مستوى التطبيق في قدرة الأجهزة الفاعلة على النقل أثناء استعادة الاتصال بالشبكة. بالإضافة إلى النظر في متطلبات المستوى المادي، تأخذ C^2AM أيضا بعين الاعتبار مخاوف مستوى التطبيق من أجل تجنب اضطرابات كبيرة في البعثات الجارية. قد قامت نتائج المحاكاة بالتحقق من صحة فعالية الخوارزمية المقترحة في الحفاظ على هذين الهدفين. علاوة على ذلك، نقوم بالتحقيق في مشكلة استعادة الاتصال المقيدة بقيود الاتصالات لطول المسارات بين الأجهزة الفاعلة من أجل التعامل مع تأخر البيانات واحتمال فقدان بعض حزم البيانات. نقتح خوارزمية للإصلاح باستخدام البنية الأقل تخريبا (LeDiR). على عكس المخططات المعاصرة التي تحافظ على قوائم 1 أو 2 - هوب للجيران، تقوم LeDiR بالانتفاع من أنشطة اكتشاف المسار السابقة في الشبكة لمعرفة هيكل الطوبولوجيا والابتعاد عن فرض أحمال إضافية للاتصالات القائمة قبل الفشل. تم تحليل أداء LeDiR رياضيا والتحقق من صحتها من خلال تجارب المحاكاة واسعة النطاق. لقد قمنا بتوسعة LeDiR وتسميته بخوارزمية إصلاح الطوبولوجيا بأقل عدد من الحركات (LeMoToR). مثل مخطط LeDiR، فإن LeMoToR هو مخطط موزع يعتمد على العرض المحلي المقدم من عقد للشبكة. ومع ذلك، فإن LeMoToR لا يفرض أي قيود للحفاظ على طول الطريق بين أي زوج من العقد في حالة قبل الفشل ويطبق نفسه بشكل متكرر للحفاظ على الاتصال بالشبكة. لاستعادة الاتصال تسعى LeMoToR لنقل أقل عدد من العقد وتقليل المسافة المقطوعة وتقليل تعقيد الرسالة. قامت نتائج المحاكاة بالتحقق من فعالية LeMoToR.

درجة الماجستير في العلوم

جامعة الملك فهد للبترول والمعادن

الظهران، بالمملكة العربية السعودية

CHAPTER 1

INTRODUCTION

Wireless Sensor-Actor Networks (WSANs) have attracted lots of interest in recent years. WSANs can increase the effectiveness of numerous applications such as homeland security, battlefield reconnaissance, space exploration, search and rescue, etc. A typical WSAN consists of a larger set of miniaturized sensor nodes reporting their data to significantly fewer actor (actuator) nodes [1][2][3][4][5][6][7][8]. Sensors probe their surroundings and report their findings to one or multiple actors, which process the collected sensor reports and respond to emerging events of interest. An actor's response would depend on its capabilities, which varies based on the application and the expected role the actor plays. For example, an actor can deactivate a landmine, extinguish a fire and rescue a trapped survivor. It is worth noting that a heterogeneous set of actors may be employed and assigned complementary roles.

In most application setups actors need to coordinate with each other in order to share and process the sensors' data, plan an optimal response and pick the most appropriate subset of actors for executing such a plan. For example in forest monitoring applications, actors

such as fire trucks and flying aircrafts need to collaborate with each other in order to effectively control a fire and prevent it from spreading. The selection of actors that need to be engaged can be based on many factors such as actor's capabilities, actor's proximity to the detected event and actor's current load. All of these factors would require a frequent update of the actor's state. To enable such interactions, actors need to stay reachable to each other. In other words, a connected inter-actor network has to be maintained at all time.

In the following section we provide a brief overview of network connectivity issue for WSANs. Section 1.2 provides a summary of the contribution of thesis. System model has been explained in section 1.3 and section 1.4 discusses problem formulation and methodology. Organization of the rest of the thesis is stated under section 1.5.

1.1 Connectivity Issue for Wireless Sensor-Actor Networks

An actor failure can cause the loss of multiple inter-actor communication links and may partition the network if alternate paths among the affected actors are not available. Such a scenario will hinder the actors' collaboration and thus have very negative consequences on the WSANs application. Therefore, the actors should be able to detect and recover from the failure of one of them. Given that the WSAN usually operates autonomously and unattended, the recovery should be a self-healing process for the network and should be performed in a distributed manner. In addition, the network recovery should be both quick and lightweight. Rapid recovery is desirable in order to maintain the WSAN

responsiveness to detected events. In addition, the overhead should be minimized in order to ensure the availability of actors' resources for application-level missions.

However, actors are responsible for responding to the specific events and carry out tasks which must be consistent with the application goals [9]. Therefore unconstrained movement of actor(s) with the goal of achieving efficiency, in terms of reduced overhead, can cause a serious failure at application level. In other words, an application un-aware recovery of the inter-actor connectivity can be impractical in many scenarios. For example, consider the following scenario where an application un-aware recovery of the inter-actor connectivity can lead to a disastrous situation.

Life support medical units are unmanned robotic vehicles that are equipped with the necessary life support equipment such as oxygen tanks and masks. These actor units are deployed in an area that got hit by a natural disaster like earthquake, hurricane, etc. Human body heat sensors are also deployed all over the area. The job of these sensors is to probe the existence of a live human being in the vicinity and report it to the actors. After receiving such a report, close by actors are responsible to reach the location and provide necessary life support until the rescue team arrives. At the time when a unit (actor) is busy in providing emergency help to a survivor under the rubbles, task termination and the mobility of this unit may cause serious damage to the operation. However, after completing the operation, the unit can be mobilized to any location without constraints. Thus, a recovery mechanism is needed to determine the best connectivity restoration scheme under application level tasks termination constraints.

On the other hand, most of the recently proposed schemes found in the literature require every node to maintain partial knowledge of the network state. To avoid the excessive state-update overhead and to expedite the connectivity restoration process, these schemes rely on maintaining 1 or 2-hop neighbor lists and predetermine criteria for node's involvement in the recovery [10][11][12]. Nonetheless, 1-hop based schemes often impose high node repositioning overhead and the repaired inter-actor topology using 2-hop schemes may differ significantly from its pre-failure status.

However, some WSN applications require timely coordination among the actors. For example, during a combat operation timely interaction among actors would be required in order to accurately track and attack a fast moving target. Thus, extending the shortest path between two actors as a side effect of the recovery process would not be acceptable. Therefore, a network restoration scheme is required that must rely on the local view of a node about the network to relocate the least number of nodes and ensure that no path between any pair of affected nodes is extended relative to its pre-failure status.

In some mission critical applications, node movement is not much appreciated and moving many actor nodes as a side effect of the recovery process could lead to an application mission failure. For example, moving away number of actor nodes while busy extinguishing a fire or life supporting natural disaster victims could lead to a disaster. Hence, a recovery algorithm is needed that strives to relocate the least number of nodes and reduce the total travel distance and communication overhead.

1.2 Contribution Of Thesis

In this thesis we tackle the above stated problems. The aim is to find out a recovery mechanism that accounts for application level concerns in addition to considering physical level requirements. This is important in order to avoid major disruptions to ongoing missions. Moreover, we investigate the connectivity restoration problem subject to inter-actor communication path length constraints in order to meet the data latency requirements at the application-level. A technique to relocate the least number of nodes and reduce the traveled distance and message complexity is also considered. The contribution can be categorized as follows:

- *Connectivity restoration with application level constraints on actors' mobility:* An application un-aware recovery of the inter-actor connectivity can be impractical in many scenarios. We propose a distributed algorithm to restore inter-actor connectivity with application level constraints on actors' mobility. Unlike most of the published algorithms, our algorithm considers application level constraints on actor's mobility as a critical issue to be measured and factored in during the recovery.
- *Least disruptive topology repair:* Considering the connectivity restoration problem subject to path length constraints and guaranteeing that no data path between any pair of affected nodes is extended relative to its pre-failure status is a great technical challenge. We propose a new distributed least disruptive topology repair algorithm that restores connectivity by careful repositioning of nodes. Our proposed algorithm re-establishes network connectivity after node failure and does not extend the length of any data path. It totally relies on the local view of the network and does not impose any pre-failure overhead.

- *Least movement topology repair*: Relocating the least number of actors to re-establish network connectivity after failure while utilizing existing path discovery activities to get and maintain topology related information and imposing no additional pre-failure communication overhead is very challenging. We propose a least movement topology repair distributed algorithm that relies on the local view of a node about the network. It actually is an extension to LeDiR. To restore connectivity, our proposed algorithm strives to relocate the least number of nodes and reduce the traveled distance and message complexity.

1.3 System Model

A WSN involves two types of nodes: sensors and actors. Sensors are inexpensive, highly energy-constrained and having limited data processing capabilities. On the other hand, actors are generally moveable and more capable nodes with relatively more onboard energy supply and richer computation and communication resources. The transmission range of actors is finite and significantly less than the dimensions of the deployment area. Although, actors theoretically can reach each other via a satellite channel, frequent inter-actor interaction as required by WSNs applications would make the often-intermittent satellite links power-consuming and unsuitable. It is thus necessary for actors to rely mostly on contemporary inter-actor wireless links for coordination among themselves. The communication range of an actor refers to the maximum Euclidean distance that its radio can reach. Meanwhile, the action range of an actor is defined as how far it can be effective from its current position.

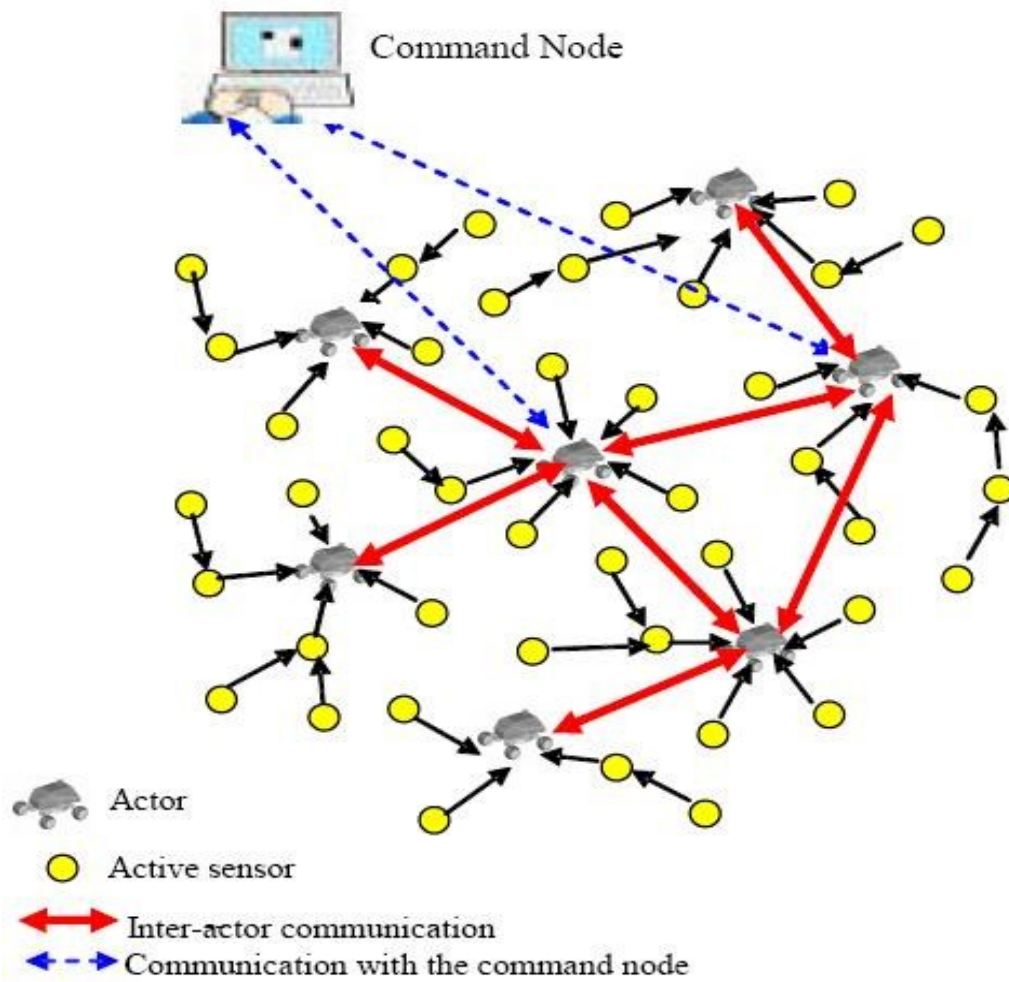


Figure 1.1: An articulation of a WASN with a connected inter-actor network.

Actors are assumed to be randomly deployed in an area of interest. In this work, network topology is considered to be a flat graph which is 1-connected and bi-directional. Upon deployment, actors are assumed to discover each other and form a 1-connected network using some of the existing techniques, such as [22]. An actor employs ranging technologies and localization techniques in order to determine its position relative to its neighbor [13]. We assume that the actors can move on demand in order to perform tasks on larger areas or to enhance the inter-actor connectivity. Given the application-based interaction, an actor is assumed to know how many actors are there in the network. Figure 1.1 articulates the considered WSN model. An actor collects sensors data in its neighborhood and collaborates with other actors. Some of the actors can interact with a remote command center through a long haul communication link, e.g., through a satellite, to report on their activities and detected event/targets. It is worth noting that although we consider such a system model, our algorithms are also applicable to mobile robotic networks where no sensors are employed.

As mentioned earlier, the focus of this work is on restoring strong connectivity at the level of inter-actor topology. It is assumed that a sensor node can reach at least one actor over multi-hop paths and will not be affected if the actors have to change their positions. Thus, sensor nodes are not part of recovery process. In the balance of the thesis, actor and node are used interchangeably. In addition, we assume that only non-simultaneous node failures will take place in the network. This, nonetheless, is not a limitation for our work. To the best of our knowledge, most recovery schemes found in the literature assumes no simultaneous faults. The rationale is that the probability for having multiple simultaneous

failures is very small. If “ p ” is the probability for a node failure, the probability for 2 simultaneous faults is p^2 and, p^3 for 3, etc. With p being a small fraction, the probability of multiple faults diminishes. In addition, the presentation of our work focuses on the algorithmic part of the recovery without focusing on the link layer issues. In general, any distributed medium access arbitration scheme would suffice.

1.4 Problem Formulation & Methodology

This work investigates means for restoring the connectivity of an inter-actor network that got partitioned due to the failure of a critical actor, i.e., cut-vertex node. In the following subsections, we define three different topology repair problems for WSANs and outline the solutions.

1.4.1 Topology Repair With Application Level Constraints On Mobility

Problem Definition: Actors in WSANs can move in various situations. However, these movements should not only be decided at the physical level and actors must not be free to move whenever / wherever they want. There must be some constraints on the repositioning of actors e.g., current task, delay bound and clustering issues. Keeping in mind such restrictions or at least application level task involvement restrictions on actors’ movement, restoring inter-actor connectivity can be a challenging issue if an actor failure causes network partitioning. Figure 1.2-(a) presents a 1-connected inter-actor network topology. In this topology a non-critical actor failure such as A_2 , A_8 , and A_5 will not hurt the inter-actor connectivity as there are other alternate paths available. In addition, an actor failure at network boundary such as A_{10} , A_{15} and A_{12} , where actor’s node degree is

one, also would not damage the inter-actor connectivity. However, a cut-vertex (critical node) actor failure can partition the network. For example, failure of A_7 can partition the network into three disjoint networks as shown in Figure 1.2-(b). The same is true for A_9 and A_{13} .

Solution Overview: We propose C²AM; a distributed algorithm to restore inter-actor Connectivity with application level Constraints on Actors' Mobility. We define two new indices: Mobility Readiness Index (MRI) and Mobility Potential index. Every actor in the network would maintain a Mobility Readiness Index (MRI) value in the range $[0-l]$. MRI is entirely based on the importance of current task, where the stringency of actor's mobility constraint increases as value of MRI increases from 0 to l . A MRI of 0 value means the actor is free to move; while a value of l means that the actor cannot move. In addition to MRI, every actor would also maintain a Mobility Potential (MP) value. MP is defined as the number of neighboring actors which can move (i.e., $MRI < l$). Every actor would calculate its MP value by tracking its 1-hop neighbors. Neighbors will know about an actor whether it is available to move or not by checking its MRI or MP value. It is worth noting that MRI has a priority over MP. The latter would be used to break the tie, if all actors participating in the recovery process have same MRI value. Every actor periodically transmits both values along with its node degree, location, and ID to its neighbors.

Another important assumption in the deployed network topology is actors' redundancy. We assumed that most of the time there would be some available actors with MRI less

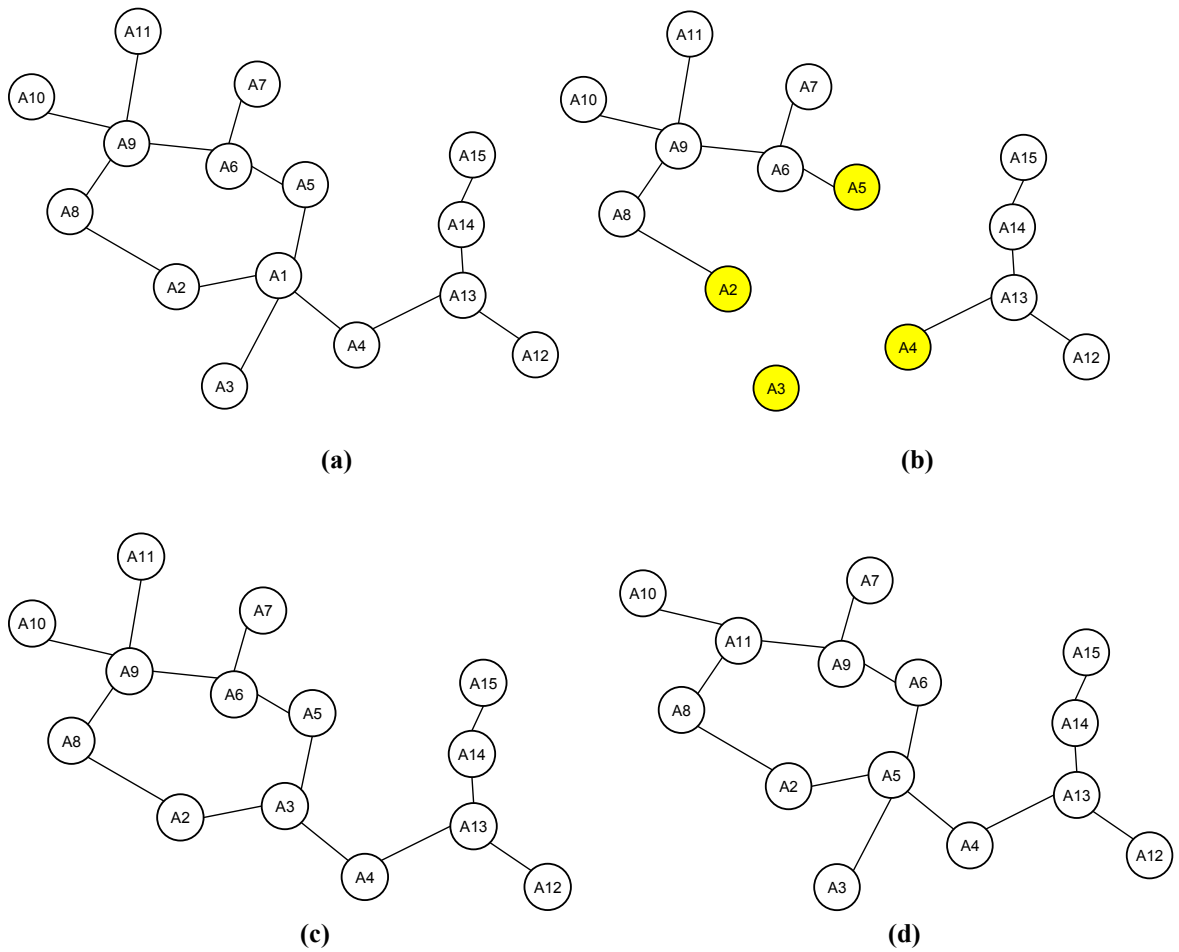


Figure 1.2: (a) Pre-failure network topology; (b) After A_1 fails the network gets partitioned into three disjoint sub-networks; (c) By using [10], node A_3 replaced the faulty actor and reestablished connectivity between actors; (d) The topology after running C^2AM with node A_5 replacing A_1 , followed with cascaded motion of A_6 , A_9 and A_{11} .

than l in the neighborhood of a failed actor which can participate in the recovery process. A free (redundant) actor with MRI of zero is the one which is not involved in any task at the time of the failure. Special attention will be needed when no free actor is available in the neighborhood of a failed actor.

1.4.2 Least Disruptive Topology Repair

Problem Definition: There are several real WSN applications that can have very strict delay requirement and is sensitive to packet loss. Examples include combat robotic networks, search-and-rescue operation, etc. In such applications extending the shortest path between any pair (i, j) of actors as a side effect of the recovery process would not be acceptable. Therefore, considering the connectivity restoration problem subject to path length constraints is very important. The goal is to restore inter-actor connectivity in a WSN without extending the length of the shortest path among nodes compared to the pre-failure topology. The following example illustrates the importance of the effect of contemporary recovery schemes on the path length between nodes. Let's consider Figure 1.3-(a) and assume that node A_{10} fails. Connectivity restoration schemes that exploit node repositioning will replace A_{10} with one of its neighbors as shown in Figure 1.3-(b). For example, shown in Figure 1.3-(c), DARA [10] picks the neighbor with the least degree in order to limit the scope of relocation. Thus, A_{11} relocates to the position of A_{10} . The connectivity restoration process will be repeated with repositioning A_{12} to replace A_{11} , followed by relocating A_2 to where A_{12} was. Finally, A_{13} replaces A_2 . The resulting topology is shown in Figure 1.3-(d). While A_0 and A_3 were directly reachable to A_2 before the failure, the repaired topology in Figure 1.3-(d) makes the shortest path one hop longer

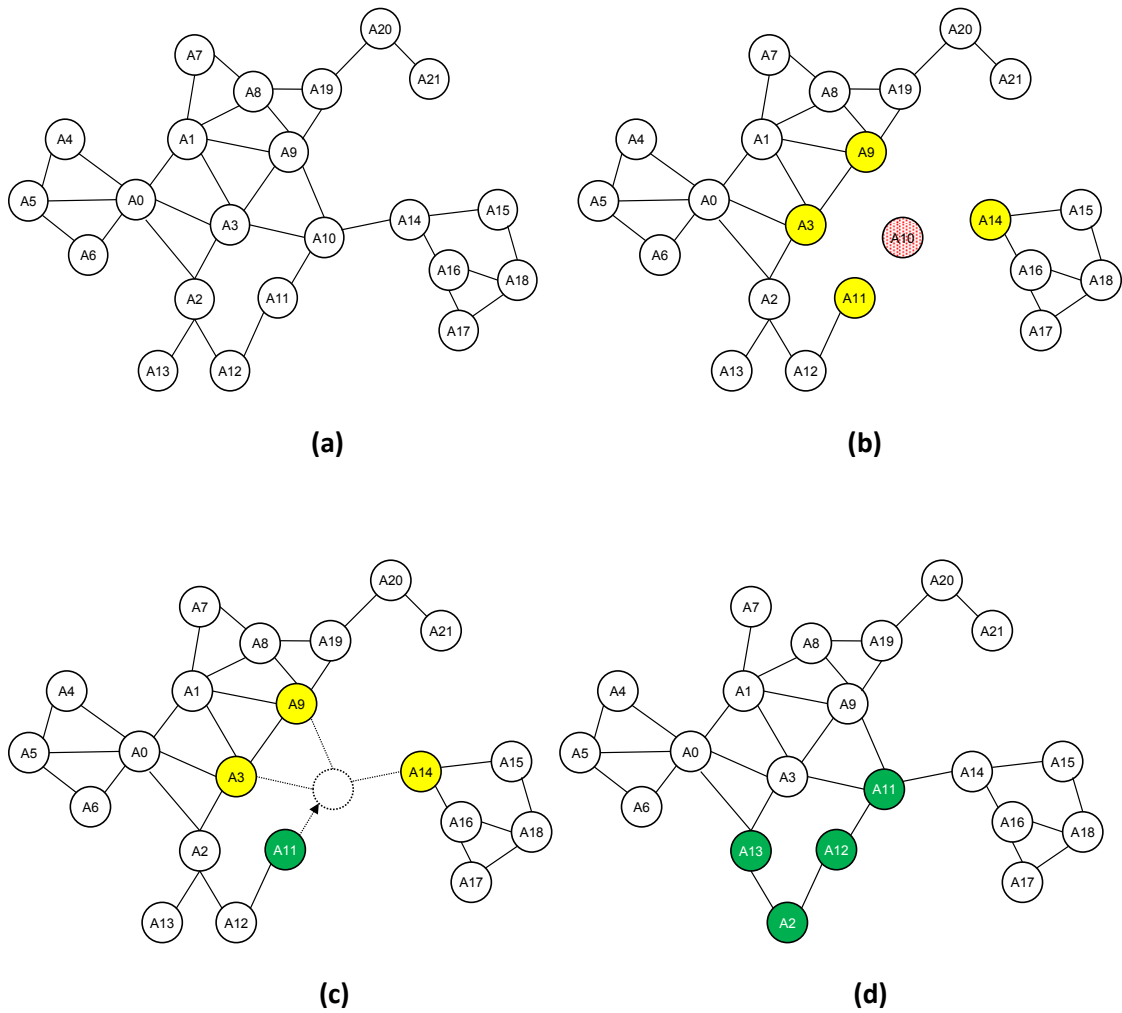


Figure 1.3: Illustration on how DARA [10] restores connectivity (a) Initial 1-connected WSN topology (b) Disjointed network with faulty node A_{10} and potential best candidates A_3 , A_9 , A_{11} , and A_{14} (c) Based on least node degree, node A_{11} has been selected as best candidate to replace the faulty node A_{10} (d) Repaired topology with the highlighted nodes A_{11} , A_{12} , A_2 and A_{13} that participated in the recovery process.

by involving A_{13} . As mentioned above, this will not be acceptable for delay sensitive applications. Moreover, increase in path length boosts the potential of packet loss. Thus, such scenario must be avoided by sustaining or even shortening the pre-failure path lengths.

Solution Overview: We propose a novel Least-Disruptive topology Repair (LeDiR) algorithm. LeDiR relies on the local view of a node about the network to relocate the least number of nodes and ensure that no path between any pair (i, j) of affected nodes is extended relative to its pre-failure status. LeDiR is a localized and distributed algorithm that leverages existing route discovery activities in the network and imposes no additional pre-failure communication overhead.

To simplify the presentation, a centralized implementation of LeDiR is assumed, where every node is aware of the entire network topology prior to the failure and thus can build the shortest-path routing table (SRT) for every pair (i, j) of nodes. This assumption is eliminated later. LeDiR is a distributed scheme that does not need a network-wide state. The SRT can be populated through the route discovery activities in the network, e.g., when an on-demand routing protocol such as AODV is employed.

The main idea for LeDiR is to pursue block movement instead of individual nodes in cascade. In order to limit the recovery overhead, in terms of the distance that the nodes collectively travel, LeDiR identifies the smallest among the disjoint blocks. When a node fails, its neighbors will individually consult their possibly-incomplete SRT to decide on

the appropriate course of action and define their role in the recovery if any. If the failed node is a cut-vertex (critical node), i.e., a node that causes network to partition into disjoint blocks, the neighbor that belongs to the smallest block reacts. For the previous example when A_{10} fails, LeDiR will only involve the block of node A_{14} . In addition, LeDiR opts to limit the travel distance by stretching the links and moving a node only when it becomes unreachable to their neighbor.

1.4.3 Least Movement Topology Repair

Problem Definition: During the network restoration process, overall high node movement can have negative effect on movement-sensitive applications. Let's consider Figure 1.3-(a) and assume that node A_{10} fails. Some connectivity restoration schemes that exploit node repositioning will recover the network by involving the neighbors of A_{10} . For example, RIM [11] picks the 1-hop neighbors and moves them to $r/2$ unit away from the faulty node A_{10} . Thus, A_3 , A_9 , A_{11} and A_{14} relocate to the new positions which are $r/2$ unit away from A_{10} and strongly reconnect the network. However, the connectivity restoration process triggers further relocations of the neighbors (children) of each moved node. The resulting topology is shown in Figure 1.4. Highlighted nodes are moved and somehow get involved in the recovery process. This will not be acceptable for movement-sensitive applications. Thus, such scenarios require least possible node movements while restoring network connectivity. Moreover, confining the node movement within the smallest portion of the network is desirable.

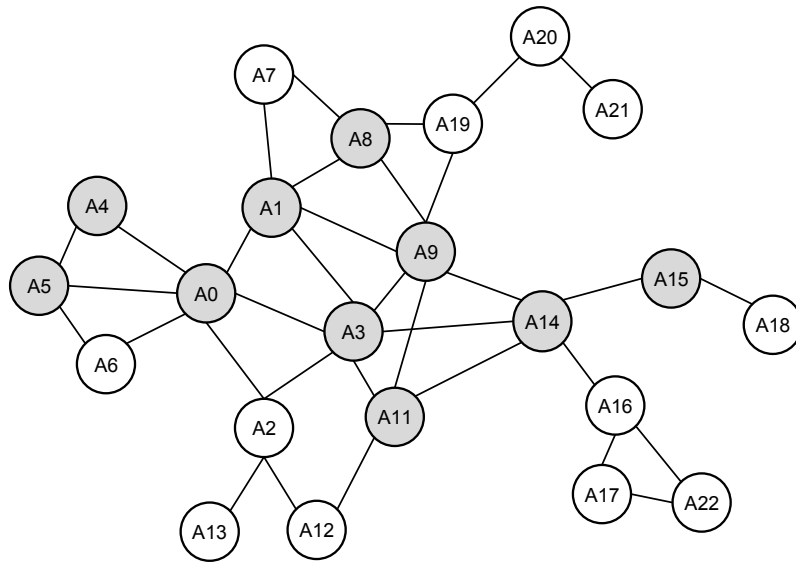


Figure 1.4: Illustrating how RIM [11] restores connectivity after the failure of node A_{10} in the connected inter-actor topology of Figure 1.3-(a). Highlighted nodes are moved and get involved in the recovery process.

Solution Overview: We extend our LeDiR algorithm and name it Least-Movement Topology Repair (LeMoToR). Like LeDiR algorithm, LeMoToR replaces the faulty node by selecting a neighbor node that belongs to the smallest disjointed block. However, LeMoToR is applied further recursively in case the node replacing the faulty node gets disconnected from its children i.e., neighbors within the block. This will not only move the least number of actor nodes but also limit the recovery overhead in terms of the distance that the nodes collectively travels. For the previous example when A_{10} fails, LeMoToR will only involve the block of node A_{14} . In addition, LeMoToR opts to avoid the effect of the relocation on coverage by moving a node only when it becomes unreachable to their neighbor. We assume that every node is aware of the entire network topology prior to the failure and thus can build the SRT for every pair of nodes. However, this assumption is eliminated later. Without loss of generality, hop count is used to calculate the inter-actor path cost.

1.5 Organization Of Thesis

The rest of this thesis is organized as follows. Chapter 2 surveys the related work. In Chapter 3, we report on the simulation tools and environment. Also, different performance metrics are discussed that are used in the simulation experiments to validate our proposed algorithms. Chapter 4 covers our new distributed algorithm C²AM. Chapter 5 presents our novel LeDiR algorithm that overcomes the shortcomings of contemporary recovery schemes which either impose high node relocation overhead or extend some of the inter-actor data paths. Chapter 6 discusses an extension of LeDiR algorithm that we called LeMoToR. Chapter 7 concludes the thesis and discusses the future work.

CHAPTER 2

LITERATURE REVIEW

In WSN, actors can move for any valid reason such as enhancing the network converge/connectivity or recovering from network partitioning. In [8], these movements are well surveyed and categorized into two major classes: initial deployment/application startup (post-deployment) and movement at any time (on-demand). Post-deployment movements are not the concern in this thesis since they are usually part of the network setup procedure. Nonetheless, most of these approaches focus on maintaining the network connectivity or enhancing sensor coverage. Little has been done to consider the application requirements or the application mission while striving to meet their main goals. In fact, considering the application requirements in the network topology optimization may introduce resource conflicts and cause deadlock. In addition, while some schemes recover the network by repositioning the existing nodes, there exists others schemes that carefully place additional relay nodes[33][34][35][36].

On the other hand, some work on sensor relocation focuses on metrics other than connectivity which is not our focus in this thesis. Table 2.1 highlight such node relocation schemes along with their performance metrics.

TABLE 2.1: Node relocation schemes that use metrics other than connectivity

S. No.	Node Relocation Scheme(s)	Performance Metric(s)
1	[14][15][16][17][18]	Coverage
2	[19]	Network longevity
3	[20]	Asset safety
4	[21][22][23]	Self-spread the nodes after non-uniform deployment

2.1 Recovery Through Node Repositioning

The main idea of this category of recovery schemes is to reposition some of the healthy nodes in the network in order to reinstate strong connectivity. Our work fits in this category. Some recent work, e.g., [24][25], have considered the application requirements when reconstructing the network topology. In [24], deadlock avoidance algorithms are proposed to tackle the challenge of sharing resources among mobile sensors with multiple missions. Meanwhile, the focus of [25] is on topology control for mission critical applications in static wireless ad-hoc networks with the goal of increasing the available resources for a set of mission critical applications such as high priority services in a network.

Furthermore, actors' movement can be in blocks [26] or in a cascaded fashion [10][27][28]. Block movement as defined in [26] is a solution based on the movement of all the nodes within a partition. Specifically, the neighbor of the failed node will lead the

partition and move towards the location of the failed node. During such movement, the remaining nodes in the partition maintain their current links and move in the same direction as the leader node (i.e., as a block). On the other hand, in cascaded node movement one of the neighbors of the faulty node replaces it. To maintain the connectivity, one of the children among the moved node is selected and relocates to the position of moved node. This process continues until every child is connected.

In both cases, block and cascaded, moving actor(s) which are busy with conducting a task and forcing them to terminate current task(s) would have negative or severe effect at the application level. For example, forcing a group (block) of actors which are involved in extinguishing a fire to terminate the current task and move away to maintain connectivity can have severe negative effect at application mission. On the other hand, forcing an actor at a time in a cascaded fashion to terminate an important task and move to another location to restore connectivity would also have negative effect, although it would be minor compared to the block movement. We argue that the negative effect at application level can be further minimized by considering application level constraints in addition to the physical level requirements. Other similar studies has been reported by S. Das, et al. in [30][31].

Published approaches differ in the level of involvement expected from the healthy nodes, in the required network state that needs to be maintained, and in the goal of the recovery process. For example, both DARA [10] and PADRA [12] require every node to maintain a list of their 2-hop neighbors and determine the scope of the recovery by checking

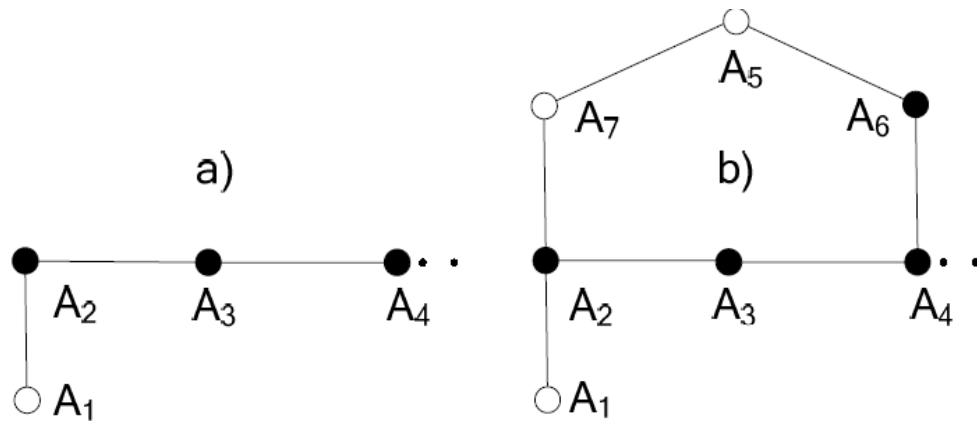


Figure 2.1: In a), actor A₁ is a dominatee and cannot be a cut vertex. A₂ is a dominator and has a dominatee A₁ which is not connected. Thus, A₂ is a cut-vertex. A₃ is also a cut vertex in a) but will not be a cut-vertex in b) [12].

whether the failed node is a cut-vertex. DARA pursues the probabilistic scheme proposed in [12] to identify cut-vertices. A best candidate (BC) is selected from the 1-hop neighbors of the dead actor as a recovery initiator and to replace the faulty node. The BC selection criterion is based on the least node degree and physical proximity to the faulty node. The relocation procedure is recursively applied to handle any disconnected children. In other words, cascaded movement is used to sustain network connectivity. On the other hand, PADRA identifies a connected dominating set to determine a dominatee node. The dominatee does not directly move to the location of the failed node, instead a cascaded motion is pursued to share the burden. In [12], also the focus is on recovering from the failure of a cut-vertex. Only a special case is considered where the failure causes the network to split into two disjoint blocks. To re-link these blocks, the closest nodes are moved towards each other. The other nodes in the blocks follow in a cascaded manner. None of these approaches cares for the path length between nodes. While some of the proposed algorithm in this work also employs cascaded relocation, the criteria for selecting the lead node and other participants are different.

In order to ensure that the recovery process converges in an efficient way, the approaches of [10][12][21] require each node in the network to be aware of its 2-hop neighbors. The availability of 2-hop list allows the nodes to detect cut-vertices with high probability and limits the scope of the recovery to cases in which the network becomes partitioned. RIM [11], on the other hand, defies that assumption and bases the recovery process on the knowledge of direct, i.e., 1-hop, neighbors. Simply the neighbors of a node “ F ” detect that “ F ” has failed and then move towards F until they can reach each other directly. Any

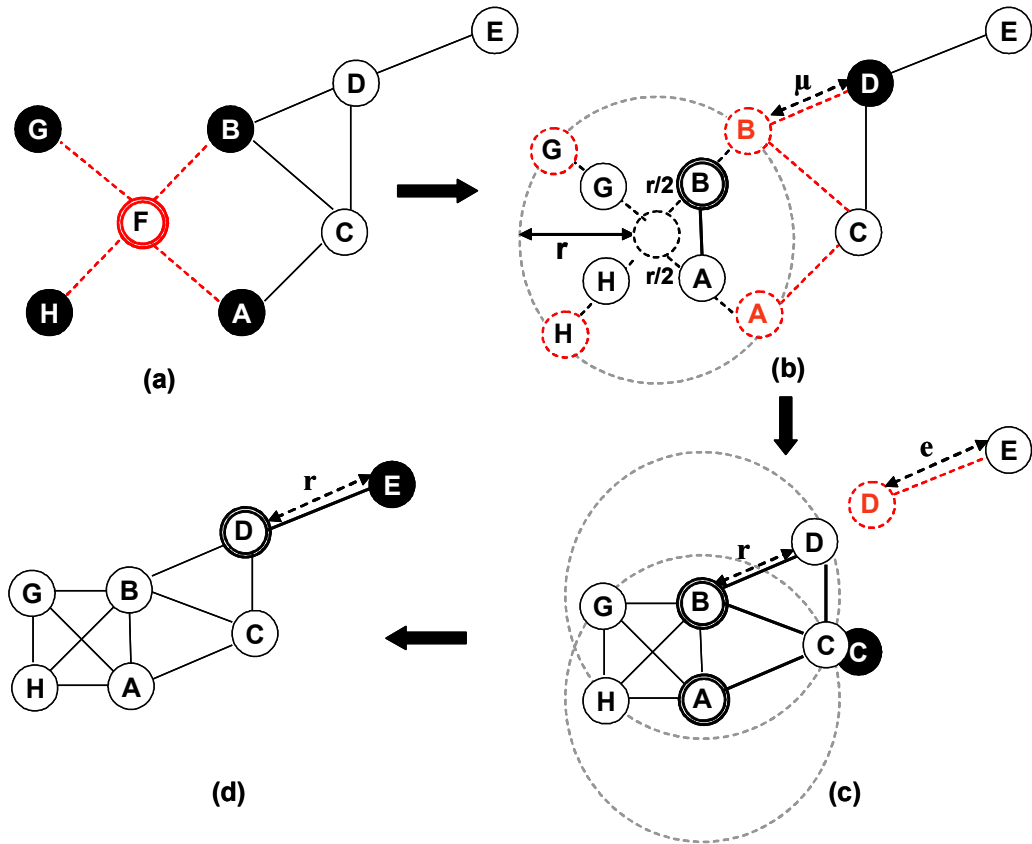


Figure 2.2: An example for how RIM restoration process; each shaded node moves based on the positions of its neighbors, denoted in double-lined circles [11].

lost link during the recovery will be reestablished through cascaded relocation. The collective effect seems like the network topology is shrinking inward. The advantage of RIM is obviously the reduced communication overhead which is nonetheless provided at the expense of overreacting to failure of nodes that are not cut-vertices. We propose to utilize the partial knowledge of a node about the network topology, gained during route discovery, to decide on which node participates and which one does not. No recovery-related explicit state update is required.

Unlike our thesis, CRR [23] avoids replacing the faulty node with a healthy node since the failure might be caused by hazards that may damage the substitute node as well. Instead, CRR rearranges the network topology in the vicinity of the faulty node. The network restoration is modeled as a Steiner tree approximation problem. A set of Steiner points are identified and the 1-hop neighbors of the faulty node are relocated at these points. In case the number of 1-hop neighbors are not enough, the approach progresses as the DARA approach, discussed above. To get a bound on performance of recovery schemes, Al-Fadhly et al. [29] formulated the problem of finding the relocation schedule with the least travel distance and maximum coverage as an integer linear program. Such a centralized approach would fit more of a planned rather a reactive recovery scenario, as targeted by our thesis.

In addition to network connectivity, coverage is also an important performance metric for WSANs. While restoring the network connectivity, coverage loss is possible either because of the failure itself or due to the connectivity-limited focus of the recovery.

Unlike the approaches discussed above, C³R [32] tackles the loss of both coverage and connectivity. C³R involves 1-hop neighbors of the faulty node in the recovery process. All the 1-hop neighbors take turn in relocating to the position of the faulty node and return back to their original position. This leads to intermittent connectivity and monitoring of all the originally covered spots. Finally, node relocation has been pursued for optimizing the network performance, including boosting connectivity, not necessarily to deal with node failure. A survey of such work can be found in [8].

2.2 Recovery By Placement Of Relay Nodes

The above algorithms aim to restore the network connectivity by efficiently relocating some of the existing nodes. However, in some setups it is not feasible to move the neighbors of the failed node due to physical, logistical and coverage constraints. Therefore, some schemes establish connectivity among the disjoint network segments by placing new nodes. The published schemes generally differ in the requirements of the newly formed topology. For example, SpiderWeb [33] and DORMs [34] opt to not only re-establish the network connectivity but also achieve a certain quality in the newly formed topology. Basically, both schemes try to avoid the introduction of cut-vertices so that some level of robustness, i.e. load balancing and high node degree, is introduced in the repaired network topology. SpiderWeb and DORMS also strive to minimize the required number of relays. Both SpiderWeb and DORMS deploy relays inwards towards the center of the deployment area. The former considers the segments situated at the perimeter and establishes a topology that resembles a spider web. Meanwhile DORMS initially forms a star topology with all segments connected through a relay placed at the

center of area. Then adjacent branches are further optimized by forming a Steiner tree for connecting two segments and the center node in order to reduce the required relay count.

Meanwhile, in [35] inter-segment connectivity ought to maintain some level of QoS while placing the least number of relay nodes. The proposed approach initially models the deployed area as a grid with equal-sized cells. Each cell is assessed based on the uncommitted capacity of the relay node residing in the cell. Finally, to meet the QoS requirement, optimization is done by finding the cell-based least cost paths and populating nodes along these paths. On the other hand, Zhang et al., [36] forms a bi-connected inter-segment topology by placing redundant nodes so that the failure of a node can be tolerated and the network operation continues without interruption.

CHAPTER 3

EXPERIMENTAL SETUP & PERFORMANCE METRICS

3.1 Introduction

This chapter describes the simulation tool, environment and performance metrics. The simulation experiments are performed on a WSN simulator developed in Visual C++. The simulator has already been validated against extensive simulation experiments as well as existing approaches in the literature. Section 3.2 provides a brief overview of WSN simulator. Network topologies are described in section 3.3 and section 3.4 provides simulation assumptions. In section 3.5, we explain the performance metrics.

3.2 Overview of WSN Simulator

In the following we provide an overview of the wireless sensor-actor network simulator.

3.2.1 General -Purpose

The simulator tool actually is a framework for general wireless sensor-actor networks. It provides an extensive framework to simulate the basic entities in the sensor-actor network. These entities are the sensor nodes, gateways (actors), packets, routes, targets etc. The basic characteristics of these entities such as communication range, action range, energy level etc. is also enumerated and a software equivalent is provided. A mechanism is provided to establish communication pathways between these pre-defined entities.

3.2.2 Design

As a whole, a typical wireless sensor-actor network consists of the following independent entities:

- Sensor nodes
- Gateways (actors)
- Clusters
- Packets
- Packet Queues
- Targets
- User-interface
- Events
- Event Queues

An object-oriented design approach is used to design the simulator where each entity is modeled by a separate object that encapsulates its functionality. These objects represent a

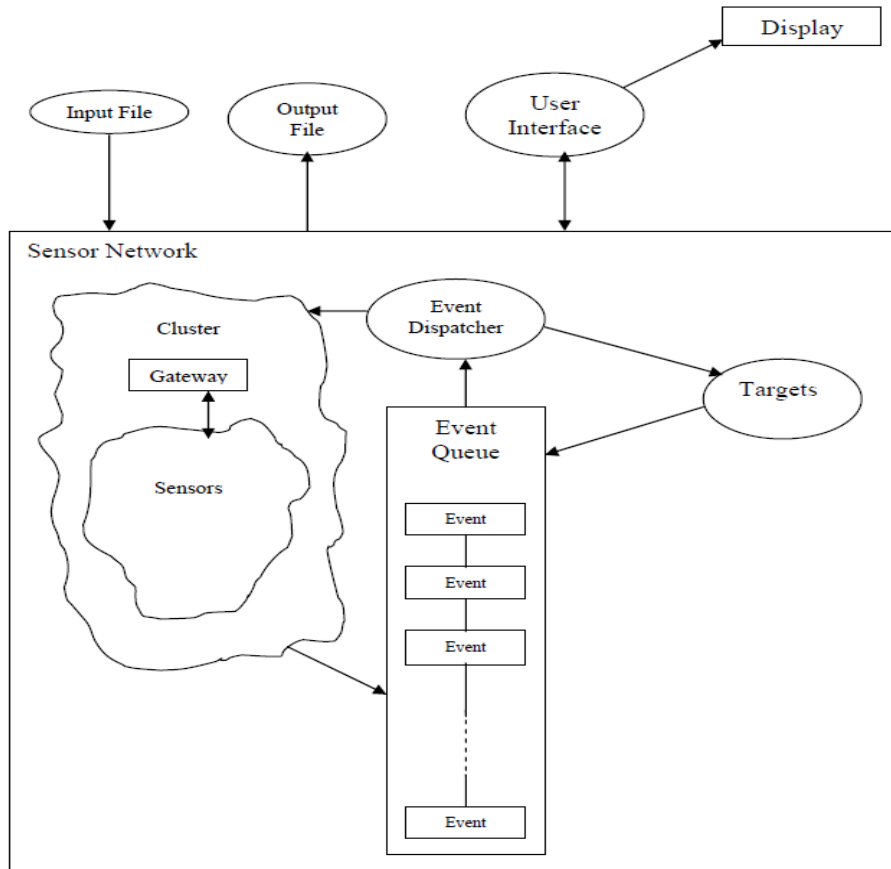


Figure 3.1: High-level block diagram of WSN simulator [42].

high level decomposition of the sensor network allowing us to establish the interactions between the entities. At a lower level, each object is assigned attributes to capture the characteristics of the entity it is encapsulating. The interactions that were established earlier are then each assigned as methods to the object [42].

3.2.3 Extensible

The WSAN simulator is very easy to extend. Actually, the simulator provides a very basic functionality of the wireless sensor-actor network. New extensions such as changes in the routing and MAC protocols, topology management algorithms, etc. can be added to the basic simulator very easily and integrate seamlessly.

3.2.4 Graphical User Interface (GUI)

An intuitive and very useful feature is GUI of the simulator. An animated display of the working of the simulator provides a valuable visual clue to the events taking place in the sensor network. Following features of WSAN can be seen on GUI:

- Positions of all the sensor nodes and gateways (actors)
- States of the nodes, e.g. whether they are turned on or off, whether they are dead.
- Communication between the nodes and gateways (actors)
- Communication routes that are established by the gateway (actors) in the network
- Inter-gateway (actor) network connectivity

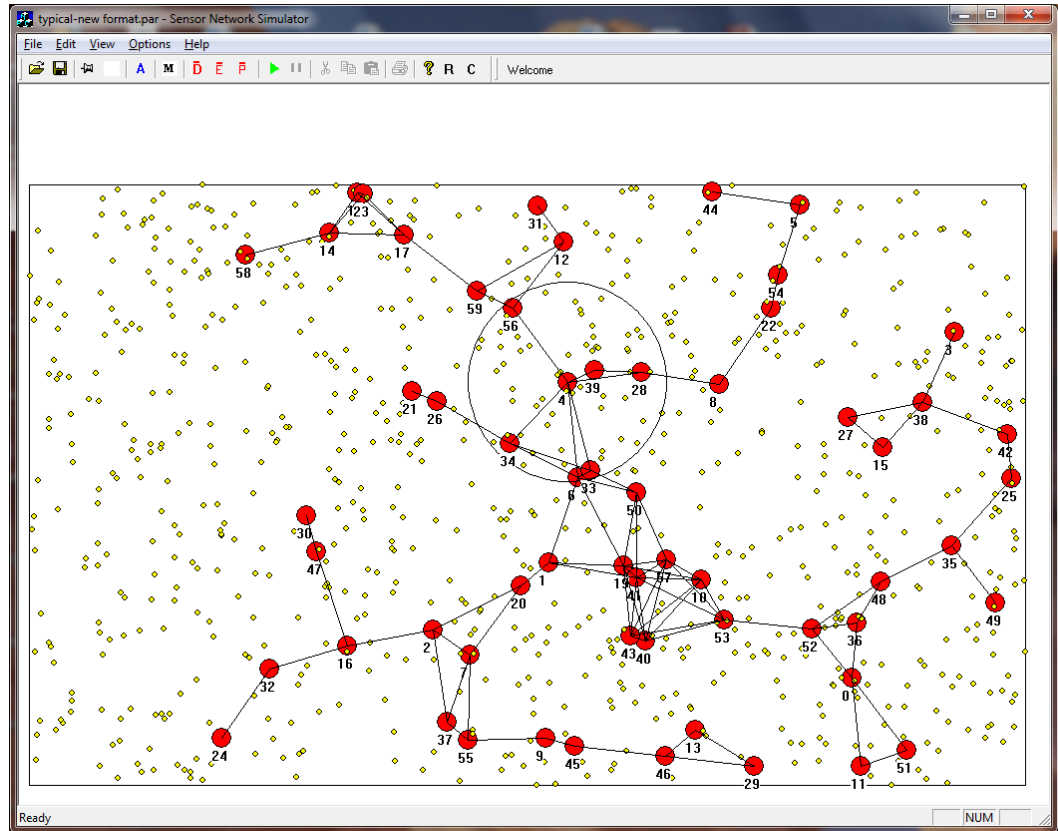


Figure 3.2: Inter-actor 1-connected network topology in WSN simulator.

3.3 Network Topologies & Environment

In the simulation experiments, 1-connected network topologies have been created. Actors are placed in an area of $1000m \times 600m$ using a uniform random distribution. For LeDiR and LoMoToR, the shortest path routing table (SRT) is formed using the Floyd-Warshall algorithm. This implicitly implies that every node is aware of the entire network topology.

Let α be the percentage of entries, i.e. routes between actor pair (i,j) , that each node has acquired over time. Hereafter, we shall call this α as Confidence Level (CL). For example, if 50% entries of node's A_i routing table are filled we say node A_i has 50% CL. We mimic the effect of Confidence Level (CL) by randomly removing $(1 - \alpha)$ % of entries from the copy of the global SRT stored at the individual nodes in order to capture the performance of a distributed implementation. All cut-vertex nodes in the topology are identified and one of them is randomly picked as the failed node and one of the proposed algorithms is applied to restore connectivity.

The following parameter is used to vary the characteristics of the WSN topology in the experiments:

- *Number of Deployed Actors (N)*: This parameter affects the node density and the WSN connectivity. Increasing the value of N makes the WSN topology highly-connected. When studying the effect of network size, the number of actors has been varied from 20 to 100 while fixing the radio range ($r = 100m$).

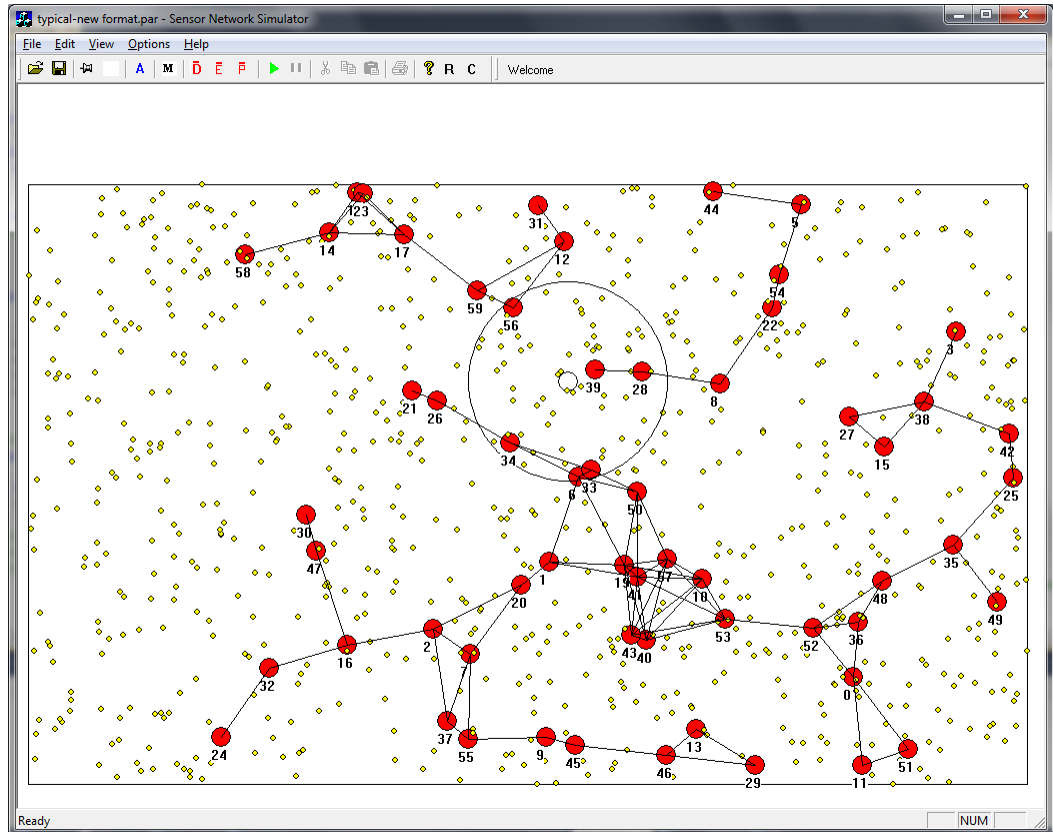


Figure 3.3: Network topology with faulty cut-vertex actor node. Topology is disjointed into 3 sub-networks.

- *Communication range (r):* All actors have the same communication range r . The value of r affects the initial WSN topology. While a small r creates a sparse topology, a large r boosts the overall network connectivity. The node count has been fixed at 100, while varying the communication range (25m to 200m).

For each simulation setup 30 different network topologies are considered and the average values are reported. We observed that with 90% confidence level, the simulation results stay within 6% - 10% of the sample mean. A detailed statistical analysis is provided in Appendix A for interested readers.

3.4 Simulation Assumptions

In the simulator, underlying physical channel in the simulation environment is considered reliable and no message loss is observed. All the nodes i.e. sensors and actors are distributed in an open space area where radio coverage is expected to be circular. The sensor and actor antenna is Omni directional. It is worth to note that the circular radio coverage assumption is widely used in literature [38][39].

As mention in Chapter 1, upon deployment, actors are assumed to discover each other and form a connected network using some of the existing techniques such as [22]. An actor employs ranging technologies and localization techniques in order to determine its position relative to its neighbor [13]. We assume that the actors can move on demand in order to perform tasks on larger areas or to enhance the inter-actor connectivity. Given the application-based inter-action interaction, an actor is assumed to know how many

actors are there in the network. Without loss of generality, all actor nodes are assumed to have the same radio / communication range which is limited and that the communication links are symmetric. However, our proposed algorithms do not require such assumption.

3.5 Performance Metrics

In order to evaluate the performance of our proposed algorithms, we quantify the overhead of the recovery process using the following two metrics:

- *Total Distance Traveled*: This metric reports sum of the distances traveled by the individual actors during the recovery. It indicates the energy incurred overhead and envisioned as a network-wide assessment of the efficiency of the applied recovery scheme.
- *Number of exchanged messages*: tracks the total number of messages that have been exchanged among nodes. This metric captures the communication-related overhead.

The following application disturbance related metric is specifically used to assess the performance of C²AM:

- *Total MRI value*: This metric captures total MRI of all actors that moved to recover the network. C2AM strives to move nodes with smaller MRI values. Thus, total MRI value is an important metric to know the degree of disturbance to the application level because of node movements. In summary, an increasing MRI value would indicate an increasing degree of disruption at application level.

The following metrics are used to validate the path length performance of LeDiR:

- *Number of shortest paths that got extended:* reports the total number of shortest paths between pairs of nodes (i, j) that get extended as a result of the movement-assisted network recovery. Please note that shortest paths are calculated by using Floyd-Warshall algorithm. This metric validates our claim that LeDiR avoid extending any shortest path between any pair (i, j) of node while restoring connectivity. Thus, for LeDiR, this metric must be zero in all experiments.
- *Average number of shortest paths that are NOT extended per topology:* This metric assesses how serious the potential path extension concern for contemporary approaches and further validates the correctness of LeDiR (This metric should be 100% all the time for LeDiR).

In addition, the following node movement related performance metric is used in LeDiR and LeMoToR:

- *Number of relocated nodes:* reports the number of nodes that moved during the recovery. This metric assesses the impact of the restoration algorithm on the ongoing activities by other actors as well as the scope of the connectivity restoration within the network.

CHAPTER 4

CONNECTIVITY RESTORATION WITH APPLICATION MOBILITY CONSTRAINTS

4.1 Introduction

A WSN may get partitioned into disjoint segments, if a critical actor, i.e., a cut-vertex node, fails and causes the loss of multiple inter-actor communication links. In such a case inter-actor collaboration would not be possible and most probably cause a fatal error/failure to the entire application mission. Since WSN applications work autonomously and unattended, actors must have a quick, lightweight, self-healing and localized mechanism to deal with such a situation. Actors are responsible for responding to the specific events and carry out tasks which must be consistent with the application goals. Therefore unconstrained movement of actor(s) with the goal of achieving efficiency, in terms of reduced overhead, can cause a serious failure at application level. In other

words, an application un-aware recovery of the inter-actor connectivity can be impractical in many scenarios.

This chapter provides the technical details and performance evaluation of our C²AM algorithm. Next section describes the steps of C²AM algorithm in detail. Section 4.3 provides detailed worked-out examples on application-aware recovery. Pseudo code of C²AM is explained in section 4.4. Performance evaluation of C²AM is provided in section 4.5. Section 4.6 provides concluding remarks on C²AM.

4.2 Detailed C²AM Steps

C²AM is an application aware inter-actor connectivity restoration approach. It requires only 2-hop neighbor information and exploits the node's mobility in order to restore connectivity of a partitioned network. The entire recovery process progresses in a localized and distributed manner. However, each node is required to maintain a 2-hop neighbor information table, referred to thereafter as *TwoHopTable*. *TwoHopTable* allows a node to make movement-related decisions independently. The following describes the major steps of the C²AM algorithm.

4.2.1 Maintaining a List of 2-hop Neighbors

C²AM requires every actor to maintain an updated list of its neighbors. To keep the scope of the recovery local, actors store information about 1-hop and 2-hop neighbors only. To keep the list up to date, an actor will send heartbeat messages periodically to update neighborhood information to its reachable actors and to assure them about its proper

operation. Each entry in the *TwoHopTable* contains five parameters $\{Node_ID, MRI, MP, Node\ Degree, Relative\ position\}$, where *Node_ID* is a unique identifier for an actor at the network level. The information stored in *TwoHopTable* is critical for the successful network recovery since it allows a node to know which actor is the most qualified to perform the recovery. A node that has passed the qualification test would be considered as the most suitable replacement of the failed node. We shall thereafter refer to such an actor as A_{Passed} . The *TwoHopTable* would be updated immediately after A_{Passed} has reached to its new location. In addition, an actor that intends to change its position will inform its neighbors beforehand in order to avoid being wrongfully perceived as faulty. Also, it would inform its new 1-hop neighbors by broadcasting a *HELLO* message as soon as it arrives at its new location.

4.2.2 Detecting a Failure and Initiating the Recovery Process

To detect a failure, C^2AM watches for repeated misses of the heartbeat messages in order to avoid overreacting to occasional packet losses over the wireless medium and to make sure that all neighbors of the failed node has a consistent assessment about A_f . When a failure is detected, decision whether to activate recovery depends on the position of the failed actor's in the network topology. Execution of C^2AM will be triggered only if a critical node, i.e., cut-vertex, has failed. The *TwoHopTable* will be used to identify cut-vertices in the network using distributed algorithms like the one proposed in [37]. This type of algorithms generally trade off the need for a network-wide state with the accuracy of identifying cut-vertices. It has been shown that the probability of missing a cut-vertex is zero while a very high percentage of the picked nodes are really cut-vertices. For

example, using 2-hop information, it was shown that the accuracy can reach 90% [27]. Since only 1-hop neighbors of a failed node will detect and participate in the recovery process, the entire detection and recovery is categorized as a localized process. We shall thereafter refer to the failed actor as A_f .

4.2.3 Application-Aware Qualification for Movement Test

The connectivity restoration process in C^2AM involves only 1-hop neighbors of A_f . C^2AM makes sure that only a single node among 1-hop neighbors of A_f should be selected to substitute A_f . Since application level constraints on an actor are a concern for C^2AM , the challenging task is to pick a node that should not create much disturbance at the application functionality while replacing A_f . To select the most appropriate node to replace A_f , C^2AM uses the following criteria in order:

- i. Least MRI Node:* In order to minimize the total MRI, a node with least MRI value will get preference to move. A minimal total MRI would indicate that the application will be disturbed the least by the recovery process.
- ii. Highest MP value:* A node with highest MP value would imply that more 1-hop neighbors of such a node would have MRI less than l .
- iii. Least Node Degree:* C^2AM prefers to replace A_f with a neighbor that has the least node degree. Moving such a node would limit the scope of the cascaded motion.
- iv. Closest Proximity to Failed Actor:* To minimize the motion overhead, the nearest neighbor of A_f is favored.
- v. Highest Actor ID:* This would be used as a last resort to avoid the situation that could come up if two or more neighbors of A_f have identical MRI, MP, node degrees and

are equidistant to it. Thus, the actor with the highest ID will be picked to break the tie.

The actors that are involved in the recovery process, i.e., 1-hop neighbors of A_f , do not have to coordinate with each other; instead they execute C^2AM concurrently. The criteria mentioned above guarantee that only one actor would pass the qualification test and all other nodes will abandon their participation.

4.2.4 Cascaded Relocation & Algorithm Termination

Before moving to the new location, A_{Passed} notifies its 1-hop neighbors. Those neighbors that are also siblings of A_{Passed} , i.e., 1-hop neighbors of both A_{Passed} and A_f , will ignore the notification. We refer to those siblings thereafter as $siblings(A_{Passed}, A_f)$. In addition, a node that has already moved once before would ignore such notification message when received. In other words, only pure children of A_{Passed} that have not been moved before would participate in the cascaded relocation process.

A pure child that has received the notification would first delete the siblings of A_f from its *TwoHopTable* to avoid wrongly considering a sibling of A_f as a better node to move and later it would perform the node qualification test. Among the pure children of A_{Passed} , one would pass the qualification test based on exactly the same criteria used for A_f and would become the new A_{Passed} . Before moving to the new location, again the 1-hop neighbors of this new A_{Passed} at the children level would be notified. This process will continue until every child is connected or all nodes move in a cascaded manner.

4.3 Application-Aware Recovery: Examples

Upon the failure of a neighbor, an actor checks its *TwoHopTable* to find out whether there is a better candidate than itself for conducting the recovery. Since all 2-hop neighbors know about each other in advance; an actor would not pass the qualification test while there is a better alternate available for recovery.

To illustrate how C^2AM algorithm works, consider the network topology presented in Figure 1.2-(a) by assuming the attribute values in Table 4.1. It is obvious from the network topology that actor A_1 is a cut-vertex and its failure could cause the network to partition into three disjoint sub-networks. Figure 1.2-(b) depicts the situation with three sub-networks namely, $\{A_3\}$, $\{A_4, A_{12}, A_{13}, A_{14}, A_{15}\}$ and $\{A_2, A_5, A_6, A_7, A_8, A_9, A_{10}, A_{11}\}$. In a physical level based network restoration process and by utilizing only 1-hop neighbors of A_1 , nodes A_2, A_3, A_4 and A_5 would participate in the recovery process; one of them would be selected based on a specific criteria to move to the location of failed node. Among children of that node, if any, one would be selected based on the same criteria and would follow it in a cascaded fashion. For example, following the approach in [10] the actor with the least node degree is picked to replace the faulty node. Applying such a criterion to the situation presented in Figure 1.2-(b), actor A_3 will move to the location of A_1 as it is the least node degree actor among all other actors participating in the recovery process. Note that, there is not any subsequent cascaded movement since A_3 does not have children. The final topology is shown in Figure 1.2-(c).

TABLE 4.1: Attributes of the actors in Figure 1.2-(a)

Node ID	MRI	MP	Node Degree
A ₂	5	1	2
A ₃	5	0	1
A ₄	3	1	2
A ₅	1	0	2
A ₆	5	3	2
A ₇	3	0	1
A ₈	4	1	2
A ₉	3	3	3
A ₁₀	3	1	1
A ₁₁	2	1	3
A ₁₂	5	0	1
A ₁₃	5	1	3
A ₁₄	5	0	2
A ₁₅	5	0	1

However, C²AM pursues a different approach and looks first for the node with the least MRI among the 1-hop neighbors of the failed node. Note that among the 1-hop neighbors of A_1 only actor A_5 has least MRI which is 1. Thus, A_5 qualifies for replacing A_1 . Since actor A_6 is the only child of A_5 , it will move to the location of A_5 despite the fact that it has MRI of 5. Nodes A_7 and A_9 are children of A_6 and both have same MRI value of 3, thus A_9 with the higher MP value qualifies to move to the location of A_6 . Among the children of A_9 , A_{11} qualifies for moving since it has a MRI value of 2 that is lower than that of A_8 and A_{10} . Since A_{11} is a boundary node and has no children, the restoration process will terminate. Figure 1.2-(d) depicts the network after successful recovery. It is worth noting that if A_4 has MRI of zero it would be selected as a replacement of A_1 . Since A_{13} is the only child of A_4 , it simply would move to the location of A_4 . MRI and MP values of A_{12} and A_{14} are similar; therefore the node degree breaks the tie and A_{12} replaces

A_{13} . Obviously, C^2AM is a greedy heuristic and sometimes does not yield a globally optimized recovery solution. For example, moving A_3 rather than A_5 would have resulted in smaller total MRI. However, it would have needed a network-wide analysis to assess the quality of such a choice. Nonetheless, as discussed in the performance evaluation section, the simulation experiments have shown that C^2AM yields close to optimal performance.

4.4 C^2AM Pseudo Code

Figure 4.1 shows the pseudo code for C^2AM . The main procedure is outlined in lines 1-19. Basically, an actor node “ J ” will track the failure of its neighbor A_f . If node J detects a failure, it will further check whether the failed node A_f is a cut-vertex (line 2). If so, J will check whether it qualifies for moving or there exists a more suitable candidate for performing the recovery (line 3). If node J qualifies, it will move to the location of A_f after sending a movement notification message to its neighbors (line 15-16). The function “Notify_Neighbors (A_{passed} , J 's 2-hop neighbor table)” announces J 's motion, new position and 2-hop neighbor table to all J 's neighbors. Otherwise, node J checks whether it has to perform a cascaded motion (line 7). In case a node has not moved before or is not a sibling of A_f (line 8-10), it will delete the siblings of A_f from its 2-hop neighbors table and check whether it qualifies for performing the recovery (line 11-13). Deleting the siblings of A_f from 2-hop neighbors table is important to avoid confusion as those siblings of A_f have already been participating in the recovery process. If J qualifies to move (A_{passed}), it will move to the location of A_f after notifying all neighbors. A node that has performed recovery movement shall conclude by updating its 2-hop neighbor table and

1. **IF** an actor node J detects a failure of its neighbor A_f
2. **IF** neighbor A_f is a cut-vertex node
3. Initiate_QualificationTest(J); $A_{passed} \leftarrow J$
4. **ELSE**
5. Exit;
6. **END IF**
7. **ELSE IF** J receives a notification message from A_{passed}
8. **IF** $Node_J_Moved_Once$ || Sibling of A_f
9. Exit;
10. **ELSE**
11. **DELETE** siblings of A_f from J 's *TwoHopTable*;
12. Initiate_QualificationTest(J); $A_{passed} \leftarrow J$
13. **END IF**
14. **END IF**
15. Notify_Neighbors(A_{passed} , J 's *TwoHopTable*);
16. A_{passed} moves to the location of neighbor A_f
17. **UPDATE** 2-hop neighbors table;
18. $Node_A_{passed}_Moved_Once \leftarrow \text{TRUE}$;
19. Exit;

Notify_Neighbors (J , J 's *TwoHopTable*)

20. Send a message to inform about J 's motion, new position and 2-hop neighbors table to all neighbors EXCEPT that are J 's 2-hop neighbors

Figure 4.1: Pseudo code for the C^2 AM algorithm

setting a flag that it has already moved and its involvement in the recovery process is completed.

The function “Initiate_QualificationTest(J)” (line 3) is used to perform the node qualification test. According to this function, node J will not qualify to move if there is an available actor k in its 2-hop neighborhood with lower MRI value and J is connected to k via A_f . However if all 1-hop neighbors of A_f have the same MRI value, higher MP value will be used to select the best candidate. In case of a tie, a node with least node degree will be considered as a better choice to move. Again, if there is more than one actor with the same node degree, then the closest one to A_f will be selected. The node ID will be used as a last resort to break the tie.

4.5 Performance Evaluation of C^2AM

In the simulation experiments, C^2AM is compared to DARA [10], the optimal cascading approach in terms of total distance traveled and the optimal cascading in terms of least total MRI. Both optimal cascading approaches are centralized and require full and updated knowledge of entire network. The former focuses on minimizing the total traveled distance, whereas the latter provides the least degree of disturbance at the application level. Identification of cut-vertices is done immediately after generating the topology and one of the cut-vertex is selected to be faulty at random. The results of the individual experiments are averaged over 30 trials. All results are found to stay within 10% of the sample mean for a 90% confidence interval.

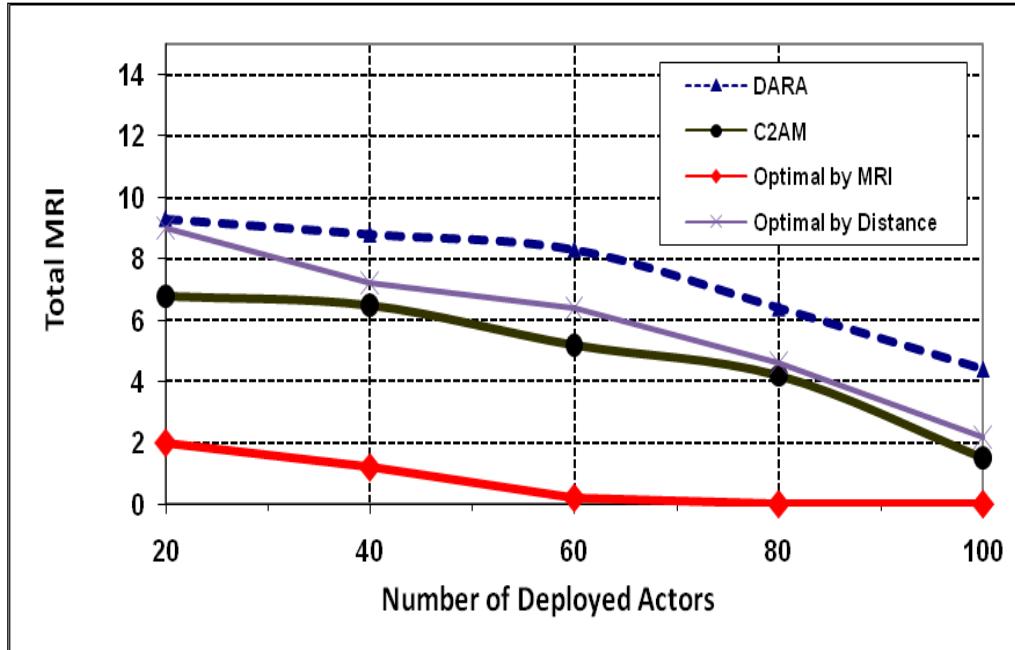


Figure 4.2: Measure of disturbance of application with varying actor count (Radio Range = 100m)

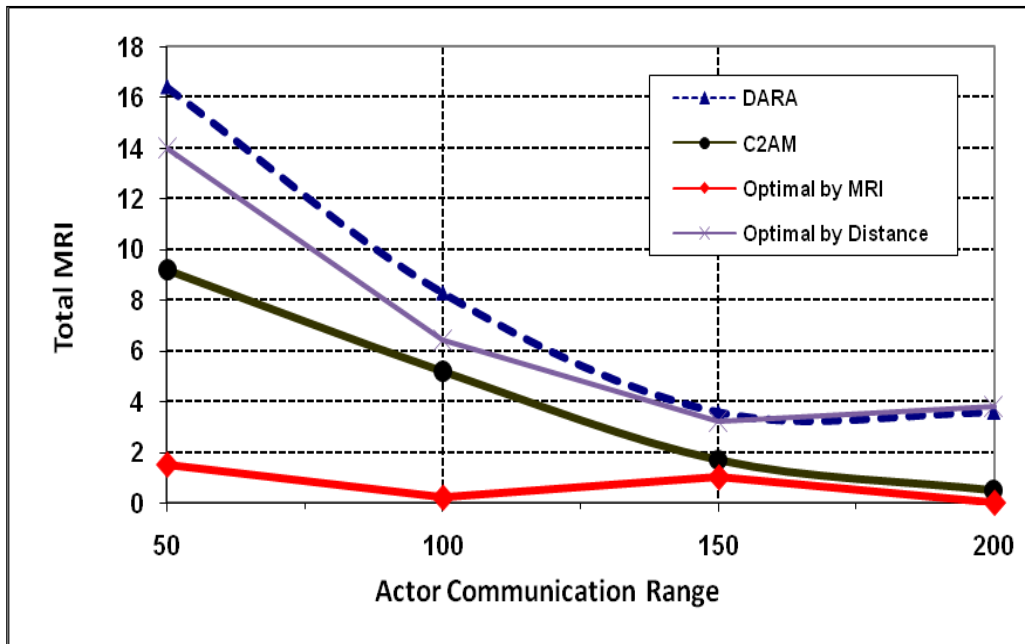


Figure 4.3: Level of disturbance to the application under varying actor radio range (with 60 actors)

4.5.1 MRI Performance

In order to assess the effectiveness of C^2AM in terms of the total MRI, we conducted experiments with varying number of actors. The results, shown in Figure 4.3, confirm the effectiveness of the C^2AM in minimizing the level of disturbance inflicted on the application as compared to the other application unaware schemes. It seems at the first glance that the performance of C^2AM is significantly less than the MRI-based optimal approach. This is mainly due to C^2AM 's concern on travel distance. In other words, C^2AM is not only caring for the application. This point will be revisited later in the section.

Figure 4.2 also indicates that the total MRI values of C^2AM get closer to those of the optimal approaches as the number of deployed actors increases. This is attributed to the fact that increasing the number of available actors would increase the connectivity and redundancy in the network. Thus, in the recovery process there would be more neighbors of a failed actor with diversified MRI values. As a result, there are higher chances that there would be more actors with small MRI values which not only would allow selecting a good candidate for replacing the failed node, but also require fewer cascaded movements to complete the restoration process. To verify our findings, we have repeated the same experiment with varying communication range of actors whereas the number of actors was fixed at 60. Increasing the actor radio range means an increase in the network connectivity. The results shown in Figure 4.3 indicate that the total MRI value decreases as the actor radio range increases, i.e., better network connectivity.

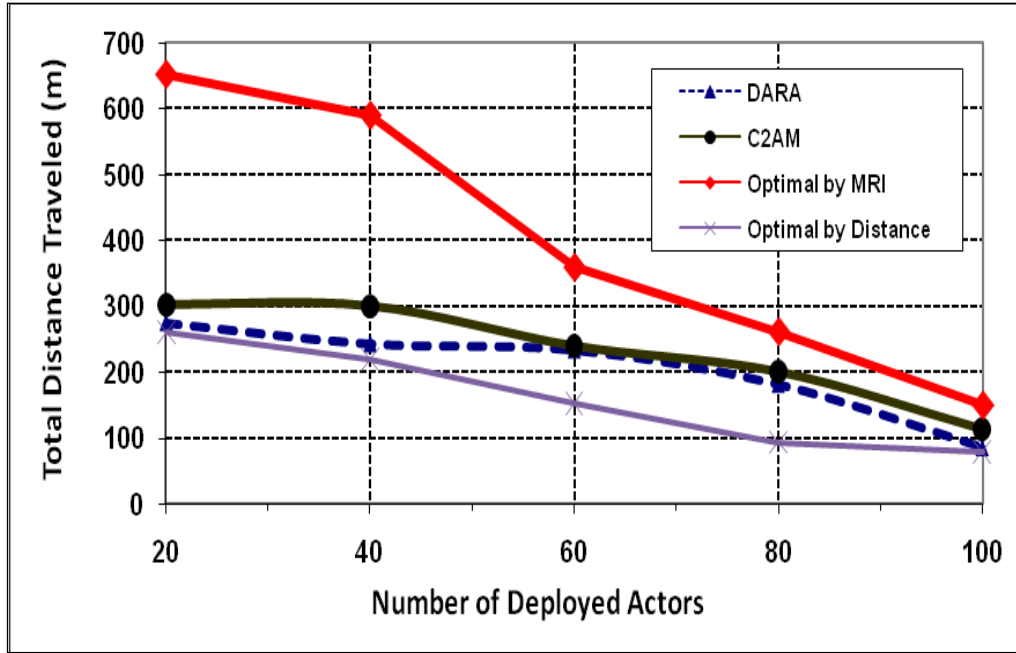


Figure 4.4: Total distance traveled with varying number of actors (Radio Range = 100m)

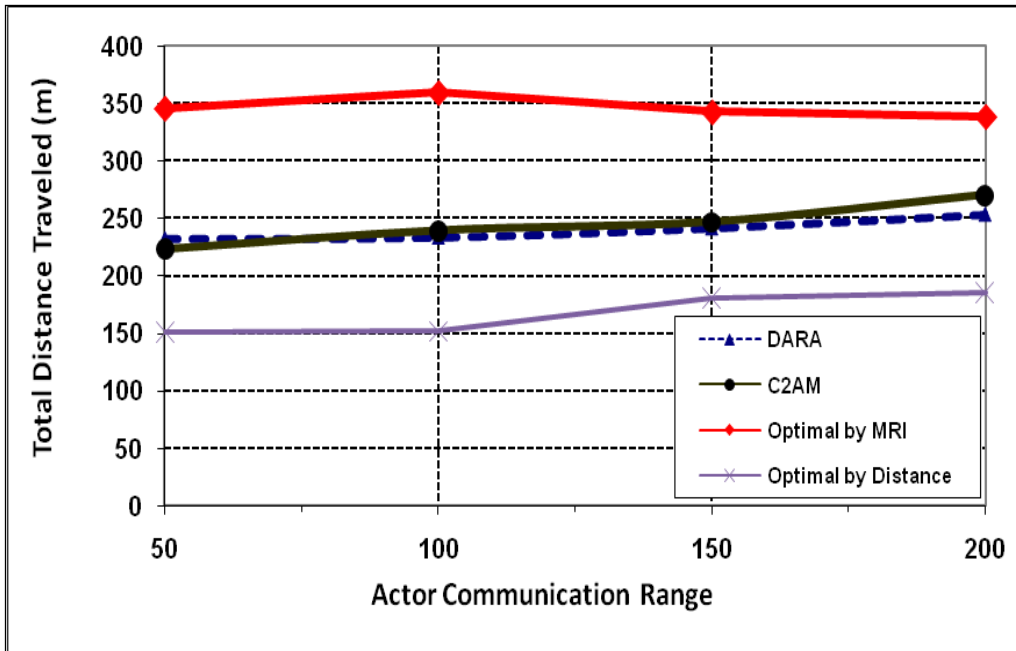


Figure 4.5: Travel distance with varying actor radio range (60 actors)

4.5.2 Movement Performance

In order to compare the movement overhead of C^2AM to DARA and the two optimal approaches, we have captured the total distance traveled with varying number of actors. The results shown in Figure 4.4 indicate clearly that C^2AM performs very close to DARA and the distance-based optimal cascading approach. As the network size grows the performance of C^2AM improves which confirms its scalability. Again, such performance is attributed to the improvement in network connectivity which limits the scope of cascaded motions. Thus, less movement is required for the recovery. The MRI-based optimal cascading approach performs significantly worse than C^2AM . When considering Figures 4.2 and 4.4 together, the results reveal that C^2AM is balancing well between keeping the total degree of disturbance at application level as low as possible and reducing the total distance traveled during the connectivity restoration. The experiments are repeated with a constant number of actors and a varying radio range. The results in Figure 4.5 also showed that C^2AM performs very close to DARA and distance-based optimal cascading approach.

4.5.3 Communication overhead

We have also recorded the total messages exchanged in the network to compare the communication overhead. Table 4.2 provides the statistics with varying number of actors with the radio range set to 100m. It can be confirmed from the table that C^2AM introduces significantly less inter-actor communication overhead than the optimal approaches. This is expected since the optimal approaches require complete knowledge of the network with each actor forced to flood the entire network that in turn produce the

TABLE 4.2: Total # of Messages Sent by C²AM with varying # of actors

# of Actors	DARA	C ² AM	Optimal Cascading	
			By MRI	By Distance
20	86.5	87.7	400	400
40	165.2	170.5	1600	1600
60	247.2	249.9	3600	3600
80	326.5	332.6	6400	6400
100	404.7	409.5	10000	10000

message complexity of $O(N^2)$. In addition, the number of messages generated by C²AM is slightly higher than DARA due to caring for the actor’s involvement in tasks. The table 4.2 also indicates that the message complexity of C²AM is linear in the network size.

4.6 Concluding Remarks on C²AM

In this chapter, we discussed a new cross-layer approach (network and application layers) to tackle the problem of connectivity repair after a node failure. The proposed C2AM approach considers two main objectives: continuous sustenance of network connectivity and minimum application level disturbance. C2AM is a localized and distributed algorithm and would thus scale well and suit the WSANs. We have validated the effectiveness of C2AM via simulation. The experimental results have demonstrated that C2AM meets both goals of minimizing the actor travel distance and communication overhead and maintaining application-level goals in a localized manner.

CHAPTER 5

CONNECTIVITY RESTORATION WITH MINIMAL TOPOLOGY CHANGES

5.1 Introduction

Given the collaborative actors' operation, a strongly connected inter-actor network topology would be required at all time. Actors usually coordinate their motion so that they stay reachable to each other. However, a failure of an actor may cause the network to partition into disjoint blocks and would thus violate such a connectivity requirement. The remote setup in which WSANs often serve makes the deployment of additional resources to replace failed actors impractical and repositioning of nodes becomes the best recovery option [8]. In addition, tolerance of node failure cannot be orchestrated through a centralized scheme given the autonomous operation of the network. On the other hand, distributed recovery will be very challenging since some nodes will not be able to reach other actors. Therefore, contemporary schemes found in the literature require every node to maintain partial knowledge of the network state. To avoid the excessive state-update overhead and to expedite the connectivity restoration process, prior work rely on

maintaining 1 or 2-hop neighbor lists and predetermine some criteria for the node's involvement in the recovery [10][11][12]. However, 1-hop based schemes often impose high node repositioning overhead and the repaired inter-actor topology using 2-hop schemes may differ significantly from its pre-failure status.

This chapter provides technical details and performance evaluation of our novel Least-Disruptive topology Repair (LeDiR) algorithm. The focus of LeDiR is on nodes that are critical to the network connectivity, e.g., cut-vertices in a graph. Uncritical nodes can be handled at the network layer of the communication protocol stack by performing topology maintenance, which may also involve node relocation [8][22]. Tolerance of uncritical nodes is usually straightforward since the network stays connected and appropriate topology adjustment can be orchestrated among the healthy nodes. The failure of critical nodes on the other hand is very challenging since the network often gets partitioned into disjoint blocks. In summary, the goal for LeDiR is to handle a critical nodes failure and restore connectivity without extending the length of the shortest path among nodes compared to the pre-failure topology. The performance of LeDiR is validated both analytically and through simulation. The simulation results demonstrate that LeDiR outperforms existing schemes in terms of communication and relocation overhead.

This chapter is organized as follows. Next section 5.2 highlights the major steps of LeDiR algorithm. Section 5.3 provides example scenarios of LeDiR and distributed implementation of LeDiR is discussed in section 5.4. Pseudo code of LeDiR is explained

in section 5.5. Algorithm analysis is provided under section 5.6. Section 5.7 discusses the performance evaluation of LeDiR and, finally, section 5.8 provides concluding remarks on LeDiR.

5.2 Major Steps of LeDiR Algorithm

We first give an overview of LeDiR as a centralized solution and then explain the distributed implementation. In the following subsections we highlight the major steps of LeDiR algorithm.

5.2.1 Failure Detection

Actors will periodically send heartbeat messages to their neighbors to ensure they are functional and also report changes to the 1-hop neighbors. Missing heartbeat messages can be used to detect the failure of actors. Once a failure is detected in the neighborhood, 1-hop neighbors of failed actor would determine the impact, i.e., whether the failed node is critical to the network connectivity. This can be done using the SRT. Basically, a cut-vertex F has to be on the shortest path between at least two neighbors of F . Consider Table 5.1 which lists the entries of the SRT for the network topology in Figure 1.3-(a). After the failure of actor A_{19} , which is a cut-vertex, node A_{20} will check what nodes are reachable through A_{19} , which are A_8 and A_9 in this example. Checking the entries for nodes A_8 and A_9 reveals that A_1 , A_3 , A_7 , and A_{10} will become consequently unreachable. The same is repeated and finally leads node A_{20} to conclude that only A_{21} is reachable and A_{19} is indeed a critical node. The SRT can make the same conclusion for a node that is not a cut-vertex but serves on the shortest path of all nodes. For example, in a wheel-

TABLE 5.1: The Path Predecessor Matrix generated by the Floyd-Warhsell algorithm [41] for the network topology of Figure 1.3-(a). For each pair of nodes v and w , the path matrix entry $P[v,w]$ contains a node k which is the direct predecessor of w on the shortest path to v .

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
0	--	0	0	0	0	0	0	1	1	1	3	10	2	2	10	14	14	16	15	8	19	20
1	1	--	0	1	0	0	0	1	1	1	3	10	2	2	10	14	14	16	15	8	19	20
2	2	0	--	2	0	0	0	1	1	3	3	12	2	2	10	14	14	16	15	9	19	20
3	3	3	3	--	0	0	0	1	1	3	3	10	2	2	10	14	14	16	15	9	19	20
4	4	0	0	0	--	4	0	1	1	1	3	10	2	2	10	14	14	16	15	8	19	20
5	5	0	0	0	5	--	5	1	1	1	3	10	2	2	10	14	14	16	15	8	19	20
6	6	0	0	0	0	6	--	1	1	1	3	10	2	2	10	14	14	16	15	8	19	20
7	1	7	0	1	0	0	0	--	7	1	3	10	2	2	10	14	14	16	15	8	19	20
8	1	8	0	1	0	0	0	8	--	8	9	10	2	2	10	14	14	16	15	8	19	20
9	1	9	3	9	0	0	0	1	9	--	9	10	2	2	10	14	14	16	15	9	19	20
10	3	3	3	10	0	0	0	1	9	10	--	10	11	2	10	14	14	16	15	9	19	20
11	2	3	12	10	0	0	0	1	9	10	11	--	11	2	10	14	14	16	15	9	19	20
12	2	0	12	2	0	0	0	1	1	10	11	12	--	2	10	14	14	16	15	9	19	20
13	2	0	13	2	0	0	0	1	1	3	3	12	2	--	10	14	14	16	15	9	19	20
14	3	3	3	10	0	0	0	1	9	10	14	10	11	2	--	14	14	16	15	9	19	20
15	3	3	2	10	0	0	0	1	9	10	14	10	11	2	15	--	14	18	15	9	19	20
16	3	3	2	10	0	0	0	1	9	10	14	10	11	2	16	14	--	16	16	9	19	20
17	3	3	2	10	0	0	0	1	9	10	14	10	11	2	16	18	17	--	17	9	19	20
18	3	3	2	10	0	0	0	1	9	10	14	10	11	2	16	18	18	18	--	9	19	20
19	1	8	3	9	0	0	0	8	19	19	9	10	2	2	10	14	14	16	15	--	19	20
20	1	8	3	9	0	0	0	8	19	19	9	10	2	2	10	14	14	16	15	20	--	20
21	1	8	3	9	0	0	0	8	19	19	9	10	2	2	10	14	14	16	15	20	21	--

shaped topology, the node at the center is not a cut-vertex, yet it serves on the shortest paths among many nodes on the outer ring. The SRT points out the criticality of such a node and motives the invocation of the recovery process.

5.2.2 Smallest Block Identification

LeDiR limits the relocation to nodes in the smallest disjoint block in order to reduce the recovery overhead. The smallest block is the one with the least number of nodes and would be identified by finding the reachable set of nodes for every 1-hop neighbor of the

failed node and then picking the set with the fewest nodes. Since a critical node will be on the shortest path of two nodes in separate blocks, the set of reachable nodes can be identified through the use of the SRT after excluding the failed node. In other words, two nodes will be connected only if they are in the same block. For example, let's again consider the network topology provided in Figure 1.3-(a) and assume node A_{19} failed. When nodes A_8 , A_9 and A_{20} the 1-hop neighbors of A_{19} confirm that A_{19} is indeed a cut-vertex (critical node), they will be able to identify the disjoint blocks. For A_{20} the analysis of the cut-vertex detection step discussed above will conclude that A_{20} can reach only A_{21} , and thus A_{20} and A_{21} constitute a block. Now, A_{20} would check the column of A_{19} and find out that A_8 and A_9 are the other direct neighbors of A_{19} . Node A_{20} will then repeat the analysis and identify the other disjoint block(s) and determine the smallest block after A_{19} fails. Now A_{20} will lead the recovery effort if it happens to belong to the smallest block, which is the case in this example. Nodes A_8 and A_9 will perform the same analysis and conclude that they are not part of the smallest block.

5.2.3 Replacing Faulty Node

If node J is the neighbor of the failed node that belongs to the smallest block, J is considered the best candidate to replace the faulty node. Since node J is considered the gateway node of the block to the failed critical node (and the rest of the network), we refer to it as “parent”. A node is “child” if 2-hops away from the failed node, “grand-child” if 3-hops away from the failed node and so on. The reason for selecting J to replace the faulty node is that moving a node and its children from the smallest block would most probably yield the least total travel distance if the entire block has to move.

As will be shown later, the overhead and convergence time of LeDiR is linear in the number of nodes, and thus engaging only the members of the smallest block will expedite the recovery and reduce the overhead. In case more than one actor fit the characteristics of a best candidate, the closest actor to the faulty node would be picked as a best candidate. Any further ties will be resolved by selecting the actor with the least node degree. Finally, least node ID would be used to resolve the tie.

5.2.4 Children Movement

When node J moves to replace the faulty node, possibly some of its children will lose direct links to it. In general we do not want this to happen since some data paths may be extended. Actually, in Figure 1.3-(d) the path between A_2 and A_3 got extended because A_2 lost its link to A_{12} after A_{12} had moved. LeDiR opts to avoid that by maintaining the existing links. Thus, if a child receives a message that the parent is moving then the child would notify its neighbors and travels directly toward the new location of parent until it reconnects with the parent again. If a child receives notifications from multiple parents it would find a location from where it can maintain connectivity to all its parent nodes by applying the procedure used in RIM [11]. Briefly, suppose a child has two parents A and B that moves, RIM relocates the child to the closest point that lies within the communication ranges of A and B . Thus, the new position of child is the closest intersection point of the two circles of radius r (which is the actor's communication range) and centered at A and B respectively. The idea also applies for more than 2 parent nodes since there must be an intersection point of 2 circles which lies within the communication ranges of all the moved nodes. It has been proven in [11] that this

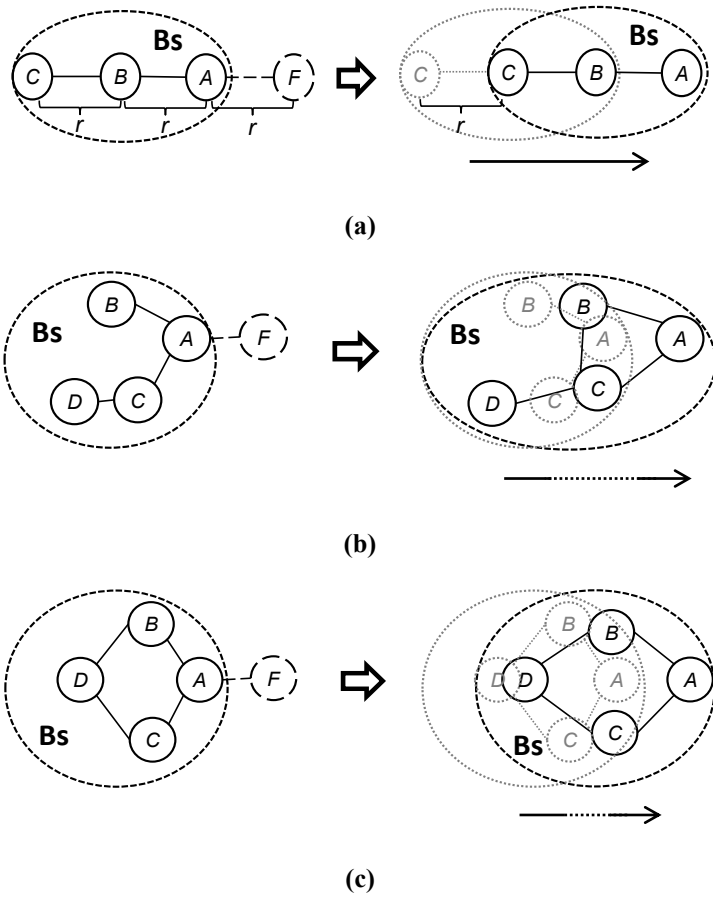


Figure 5.1: Illustrating the movement of block B_s in LeDiR to restore the network connectivity and to keep intra-block paths unchanged; (a) that entire B_s moved r units (b) the collective effect of B_s participation in the recovery is stretching B_s towards F , and (c) B_s is both stretched and moved with links within the B_s stretched in order to minimize the total travel distance. r is the actor's communication range.

relocation scheme sustains existing links in the connected component (block).

5.3 Example Scenarios of LeDiR

A simple example scenario is a 1-dimensional smallest block (B_s) where each node is r units away from each other, as presented in Figure 5.1-(a). Simply, each child would move to the location of its parent and thus the entire B_s would move r units towards node F . This would keep intra-block connectivity as is and would not extend any path within the B_s . However, in reality nodes within the B_s can be closer than r units to each other. In this scenario, movement of B_s would be performed in a way that intra-block paths remain unchanged or get shorter and total travel distance is minimized as depicted in Figure 5.1-(b). Node A moves to the location of F and children B and C get disconnected. To regain the connectivity with A , nodes B and C move towards the new location of A until becoming r units away. As a side effect, connectivity within the B_s gets stronger and creates a new link between B and C . This makes the intra-block shortest path between B and C even shorter; however, pre-movement intra-block paths remain unchanged. In addition, to avoid unnecessary movement and minimize the total travel distance, node D does not move as it is still connected to its parent C . Also, it is worth to note that the shortest path from D to B has become 1-hop shorter after recovery. Figure 5.1-(c) shows the situation where the entire B_s moves to preserve the intra-block paths with links between nodes stretched in order to minimize the total travel distance. As explained earlier, nodes B and C move toward A until r units away. Since node D has two parent that move and break their links to D , node D relocates to the closest point that lay within the communication ranges of B and C .

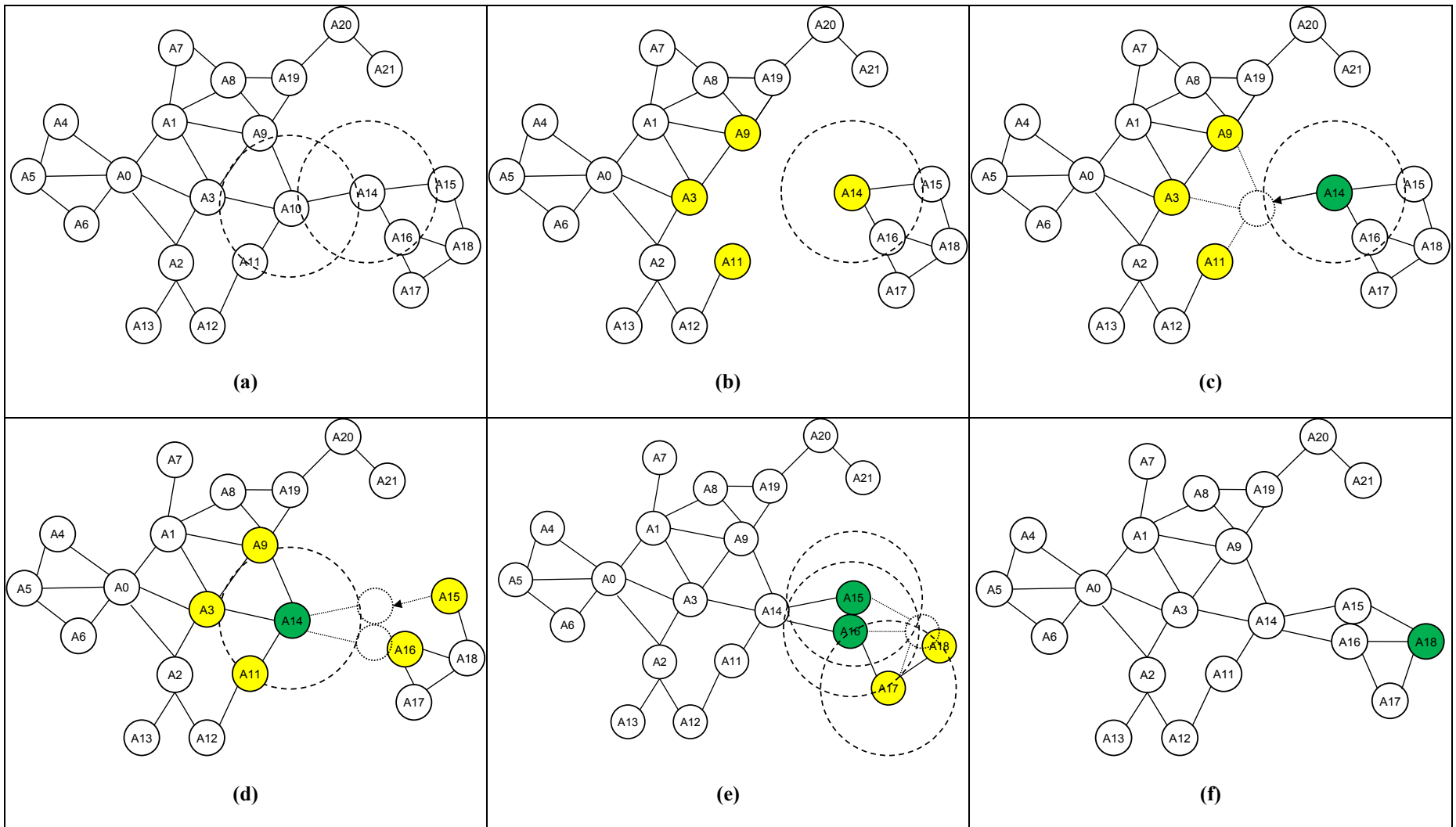


Figure 5.2: An example illustrating how LeDiR restores connectivity after the failure of node A_{10} .

Figure 5.2 shows an example for how LeDiR restores connectivity after the failure of A_{10} . Obviously, node A_{10} is a cut-vertex and A_{14} becomes the 1-hop neighbor that belongs to the smallest block (Figure 5.2-(a)-(c)). In Figure 5.2-(d), node A_{14} notifies its neighbors and moves to the position of A_{10} to restore connectivity. Disconnected children, nodes A_{15} and A_{16} , follow through to maintain the communication link with A_{14} (Figure 5.2-(e)). Note that the objective of the children movement is to avoid any changes to the current routing table. Nodes A_{15} and A_{16} would notify their children, A_{17} and A_{18} , before they move. Since A_{18} had communication links with nodes A_{15} , A_{16} and A_{17} , it moves to a new location where it can stay directly connected to these nodes (Figure 5.2-(f)). The links between A_{17} and nodes A_{16} and A_{18} are not affected by the relocation process and thus A_{17} would not need to reposition. Figure 5.2-(f) shows the repaired network topology where the paths from nodes A_{14} , A_{15} , A_{16} , A_{17} , and A_{18} to the other nodes in the network are not extended.

5.4 Distributed LeDiR Implementation

The discussion above has assumed that nodes are aware of the network topology and can assess the impact of the failure and uniquely identify which node should replace the failed actor. If every node in the network is communicating with all other nodes, it would be possible to fully populate the routing table and for the individual nodes to reach consistent decisions without centralized coordination. However, in many setups, an actor may have only partial knowledge about the network with routes to some nodes missing in its SRT. This can happen due to changes in the topology caused by node mobility or due to the fact that a subset of actors do not need to interact and simply a route has not been

discovered yet. In general, a partially populated SRT can raise the following three issues for a distributed implementation of LeDiR: (i) a potential best candidate actor does not realize that its failed neighbor is a critical node, (ii) every neighbor of the faulty node assumes that it is not part of the smallest block leaving the network topology unrepaired, (iii) more than one neighbor in different blocks step forward as best candidate. In the balance of this section we discuss how LeDiR addresses these issues.

As we mention in Chapter 3, Confidence Level (CL) is the percentage of entries, i.e. routes between actor pair (i, j) , that each node has acquired over time. Since every node may potentially have different CL from others, upon the detection of a node failure the neighboring nodes may have an inconsistent assessment of the impact of the node loss on the network and on which actor is the best candidate for leading the recovery. For example, in Figure 1.3-(a) if node A_{11} was never on a route that has nodes A_{14} , A_{15} , A_{16} , A_{17} and A_{18} as sources or destinations, node A_{11} will not know that A_{10} is a cut-vertex. We argue nonetheless that this is rare in practice since the mobility pattern among actors is not typically high given their involvement in actuation activities. In addition, the operation in WSN is collaborative in nature and an actor usually communicates with many others and thus the routing table would not be sparse. Specially, the neighbors of a cut-vertex would have more populated SRT compare to other nodes in the network as they would be passing packets among the actors in different blocks.

Furthermore, LeDiR may employ probabilistic cut-vertex detection schemes that use 2-hop information in order to boost the fidelity of the assessment [12][37]. It has been

shown that these probabilistic schemes can achieve accurate detection of cut-vertices up to 90%, i.e., no cut-vertex will be classified otherwise, and only 10% of the time a node is claimed to be a cut-vertex while it is not [12]. It is important to note that if LeDiR is applied while the failed node F turned out not to be a cut-vertex, e.g. due to the inaccuracy of the probabilistic detection scheme, the shortest path lengths between nodes will not change since LeDiR sustains the links between nodes in the same block and the network will be in fact connected, i.e., one block. Determining the block size is always based on the entries of the SRT that neighbors of F have, regardless whether F is a cut-vertex or not. Now, if the analysis to determine the block size is based on inaccurate assertion about whether F is a cut-vertex, one of the neighbors F still becomes the best candidate and performs LeDiR successfully, i.e., proceeds to replace the faulty node. Child would follow best candidate to maintain connectivity and so on.

The second and third issues above are related to determining the best candidate, i.e., the neighbor of the failed node that belongs to the smallest block. If global topological information is available, i.e., the node has a fully populated SRT, determining the smallest block is straightforward as we explained earlier. However, if a node has a low CL it may not be able to accurately determine the smallest block. For example, if node A_{14} does not have sufficient entries in its SRT it would not know that it belongs to the smallest block and would not thus initiate the recovery process by moving to replace A_{10} . Since the neighbors of A_{10} cannot reach each other, a partially populated SRT may lead to a deadlock with none of the neighbors of A_{10} responding to the failure and leaving the network disconnected. To handle this issue, LeDiR imposes a time-out after which the

neighbor(s) belonging to the second largest block will move. This time multiple neighbors may be potentially moving towards A_{10} . To avoid having more than one actor replacing A_{10} , LeDiR requires these nodes to broadcast messages with their ID so that they pause as soon as reaching other neighbors of A_{10} that happen to be in a different block. The pause time would allow these neighbors to negotiate and pick the best candidate to continue on to the position of A_{10} . We study the effect of the CL on the performance through simulation in section 5.7.

5.5 LeDiR Pseudo Code

Figure 5.3 shows the pseudo code for LeDiR. When an actor J detects the failure of a neighbor F that is considered as a cut-vertex (line 1-2); J checks its eligibility for replacing F in line 3 by consulting the SRT to find out whether it belongs to the smallest block. If J passes the test, it would move to the location of F after notifying all its children (Lines 4-10). Otherwise, node J checks whether it is to perform a movement to sustain current communication links (line 11), and if so it identifies a new position and notifies its children before moving (lines 15-20). Nodes only move once (line 12-14). The procedure *Compute_newPosition(J)* identifies where a node k (a child of J) would need to reposition based on the other notifications that it has received from nodes other than J . A new position for a node k would be computed only if k loses its direct communication link to one or multiple parent neighbors as we already mentioned in subsection 4.2.4 under children movement.

```

// Every node builds its shortest path routing table (SRT) based
// on the route discovery activities that it initiates or serve in, e.g.
// while executing a distributed routing protocol.

```

LeDiR(*J*)

```

1 IF node J detects a failure of its neighbor F
2   IF neighbor F is a critical node
3     IF IsBestCandidate(J)
4       Notify_Children(J);
5       J moves to the Position of neighbor F;
6       Moved_Once ← TRUE;
7       Broadcast(Msg('RECOVERED'));
8       Exit;
9     END IF
10  END IF
11 ELSE IF J receives (a) notification message(s) from F
12   IF Moved_Once || Received Msg('RECOVERED')
13     Exit;
14   END IF
15   NewPosition ← Compute_newPosition(J);
16   IF NewPosition != CurrentPosition(J)
17     Notify_Children(J);
18     J moves to NewPosition;
19     Moved_Once ← TRUE;
20   END IF
21 END IF

```

IsBestCandidate (*J*)

```

// Check whether J is the best candidate for tolerating the failure
22 NeighborList[] ← GetNeighbors (F) by accessing column F in SRT;
23 SmallestBlockSize ← Number of nodes in the network;
24 BestCandidate ← J;
25 FOR each node i in the NeighborList[]
    //Use the SRT after excluding the failed node to find the set of
    //reachable nodes;
26   Number of reachable nodes ← 0;
27   FOR each node k in SRT excluding i and F
28     Retrieve shortest path from i to k by using SRT;
29     IF the retrieved shortest path does not include node F
30       No. of reachable nodes ← No. of reachable nodes + 1;
31     END IF
32   END FOR
33   IF Number of reachable nodes < SmallestBlockSize
34     SmallestBlockSize ← Number of reachable nodes;
35     BestCandidate ← i;
36   END IF
37 END FOR
38 IF BestCandidate == J
39   Return TRUE;
40 ELSE
41   Return FALSE;
42 END IF

```

Figure 5.3: Pseudo code for the LeDiR algorithm

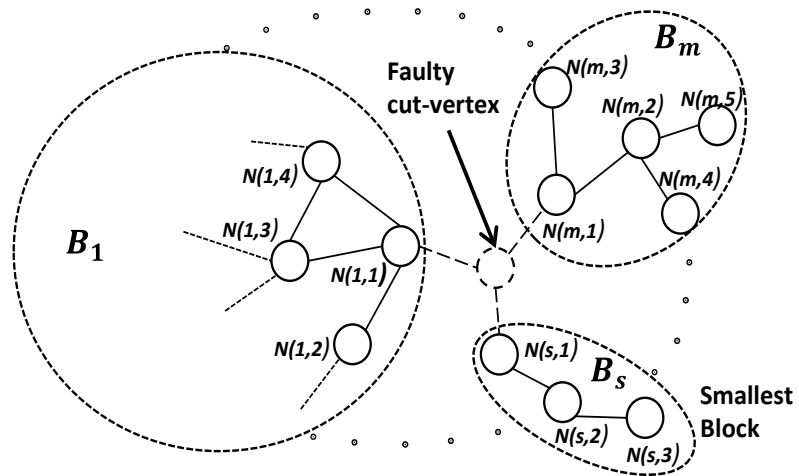
5.6 Algorithm Analysis

In this section, we prove the convergence and analyze the performance of the LeDiR algorithm. We introduce the following theorems:

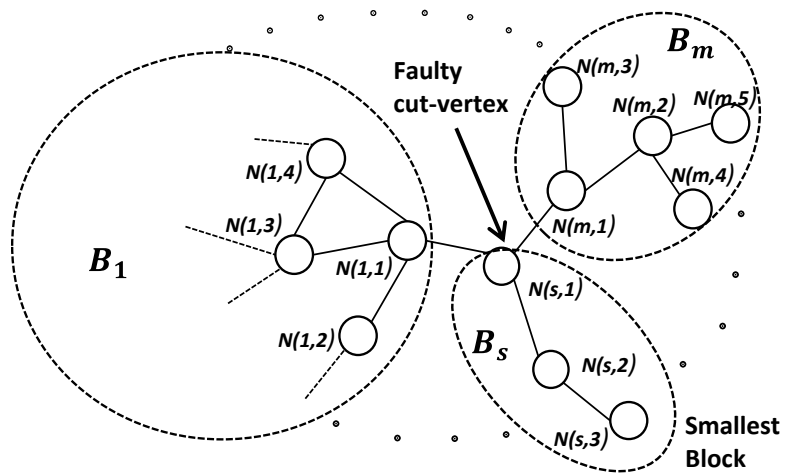
Theorem 1: “*LeDiR guarantees a localized network recovery without extending the shortest data path between any pairs of nodes (i, j) ”.*

Proof: We assume that the network is partitioned into m blocks because of a faulty node F which happens to be a critical node, e.g. a cut-vertex. The scenario is illustrated in Figure 5.4-(a), where B_s is the smallest block. The node ID is represented by $N(b, i)$ where “ b ” is block number and “ i ” is the node number within the block. LeDiR involves only 1-hop neighbors of F , denoted hereafter as $Neighbors(F)$, in the process of block selection and moves only the node in $Neighbors(F)$ that belongs to the B_s . Thus, the scope of the recovery is localized and affects only B_s .

To prove that the shortest path between any arbitrary nodes is not extended, it is sufficient to show that the inter-block paths are not extended and the intra-block paths are not longer after the recovery than before the failure takes place. Since the blocks used to reach each other through F , the node F belongs to the shortest path between every pair of nodes $N(p, i)$ and $N(q, j)$ where $p \neq q$. Thus, replacing F with a healthy node will not extend any of these paths if the intra-block part of the path is not extended. In other words, if the paths between $N(p, i)$ and $N(p, l)$, and between $N(q, j)$ and $N(q, l)$ are not extended, LeDiR will surely achieve its goal for inter-block paths. Since other than B_s none of the blocks will



(a)



(b)

Figure 5.4: LeDiR restores the network connectivity after the failure of a cut-vertex (critical node); (a) shows a WSAN before a cut-vertex fails and (b) shows the WSAN topology after applying LeDiR.

experience any changes in their intra-block topologies, the path between any pair of nodes $N(k,i)$ and $N(k,j)$ will stay intact after the recovery for all $k \neq s$. Hence, to prove the theorem it will be sufficient to prove that the shortest paths between nodes in the moved block, B_s , are not extended.

When the node $N(s,l)$ moves to replace F , the links to $Neighbors(N(s,l))$ are maintained. If a neighbor $N(s,t)$ of $N(s,l)$ was also a neighbor of F , the link between $N(s,t)$ and $N(s,l)$ is not affected. Otherwise $N(s,t)$ travels towards F to stay directly connected to $N(s,l)$. Cascaded relocation also ensures that every node stays connected to all its neighbors. To maintain pre-failure connectivity, a node which needs to move selects a new location that keeps it reachable to all its parents after the recovery. Since the motion of nodes in the B_s is inward towards F , it has the effect of shrinking the B_s towards node F and the pre-failure links of a node to its siblings are maintained. This is proven by Lemmas 1 and 2 in [11].

The analysis above shows that LeDiR not only keeps the intra-block shortest path between any pair of nodes in the B_s but also may enhance the shortest path between blocks. Since F is no longer on paths between any pair of nodes $N(p,i)$ and $N(s,j)$, some of these paths are shorter. The most intuitive example is the path between any node $N(p,i)$ and $N(s,l)$, which has replaced F . This proves that LeDiR achieves the objective to restore connectivity without extending the shortest data path between any pair of nodes in the network. \square



Figure 5.5: The worst case scenario topology where $N = 7$ and failure of A_4 has partitioned the network into two $(N-1)/2$ nodes blocks. LeDiR would involve maximum $(N-1)/2$ actors in the recovery process either A_3 or A_5 selected to replace the faulty node followed by a series of inter-block node relocation.

Theorem 2: “*The maximum number of nodes involved in the recovery process is $O(N)$, where N is the number of actors in WSAN*”.

Proof: Consider the worst case scenario where 1-dimensional network is split into two equal blocks (sub-networks) and each block consists of $(N-1)/2$ nodes, as shown in Figure 5.5. LeDiR involves only one of the two blocks in the recovery process, simply by moving only one block towards the other. Assuming that the network is sparse and nodes are r units away from each others, where r is the node’s communication range, every node in the block would move and participate in the recovery process. Thus, the maximum number of nodes involve in the recovery process $(N-1)/2$ which is $O(N)$. \square

Theorem 3: “*LeDiR strives to minimize the total travel distance and guarantees to terminate in $O(N)$ iterations, where N is the number of actors in WSAN*”.

Proof: Theorem 1 proves that LeDiR selects the smallest block for recovery which is to minimize the total travel distance by moving the smallest number of nodes. Theorem 2 proves that in the worst case scenario $O(N)$ nodes are involved in the recovery. During the entire recovery process a node can move only once which means that LeDiR guarantees to terminate in $O(N)$ iterations. \square

Theorem 4: “*The message complexity of LeDiR is $O(N)$ where N is the number of actors in WSAN*”.

Proof: LeDiR depends on the route discovery protocol to maintain the routing table on each node, thus no special messaging is required to know the neighbors or network topology. If a node got involved in the recovery process and decided to move, it broadcasts one message to its children to notify them about its movement. Another message is broadcast to interact with the neighbors once a node has reached to the new position. In other words, every node participating in the recovery process would broadcast only two messages. In the worst case scenario only $O(N)$ nodes would participate in the recovery process as proven in Theorem 3 above. Thus, the total number of messages sent is $2*(N)$ which is equal to $O(N)$. \square

Theorem 5: *“The maximum distance a node travels in LeDiR is r where r is the actor radio range”.*

Proof: In the worst case (Figure 5.6), LeDiR can select a 1-hop neighbor to replace the faulty node that is at most r units away from it. When a node moves to replace the faulty node, possibly some of its children will lose direct links to it. If a child receives a message that the parent is moving then the child would notify its neighbors and travel a maximum of r units to restore its link to the parent. If a child receives notifications from multiple parents it would find a location from where it can maintain connectivity to all its parent nodes. In this case, the new location definitely is less than r , as proven in [11].

Thus the maximum distance a node travels in LeDiR is r . \square

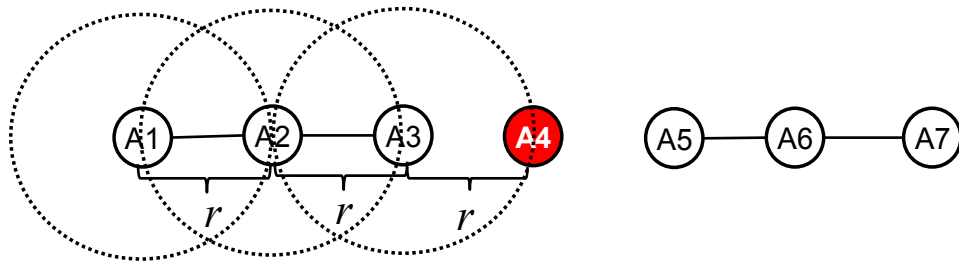


Figure 5.6: Assuming the worst case scenario presented in Figure 5.5, LeDiR selected A_3 to replace the faulty node A_4 by traveling distance r . Once A_3 moved to the new position, A_2 will move behind it to maintain direct connectivity. Later, A_1 will do the same. Since the network is 1-dimensional and nodes are located r units away from each other, the maximum distance travel by a node is r .

Theorem 6: “The maximum convergence time of LeDiR algorithm to restore inter-actor connectivity is $O(N)$ where where N is the number of actors in WSAN”.

Proof: Let’s assume that no other failure occurs during the recovery process and s is the maximum time required for a node to find whether it belongs to the smallest block. The maximum time for a neighbor “A” of the failed node “F” to find out the block that it belongs to is $O(N.d)$. Basically, a node will have to check the column for “F” in the SRT to identify all the other “ $d-1$ ” neighbors of “F”. Node “A” then eliminates these “ $d-1$ ” actors and all nodes that are reachable through them from its row in SRT. This step is applied at most $N-1$ times in a network of N actors, and node “A” is a leaf node in the network. To determine whether its block is the smallest, node “A” will repeat this process at most “ $d-1$ ” times for the other neighbors of “F”. Thus, the maximum time s for a node to identify the smallest block is $O(N.d^2)$.

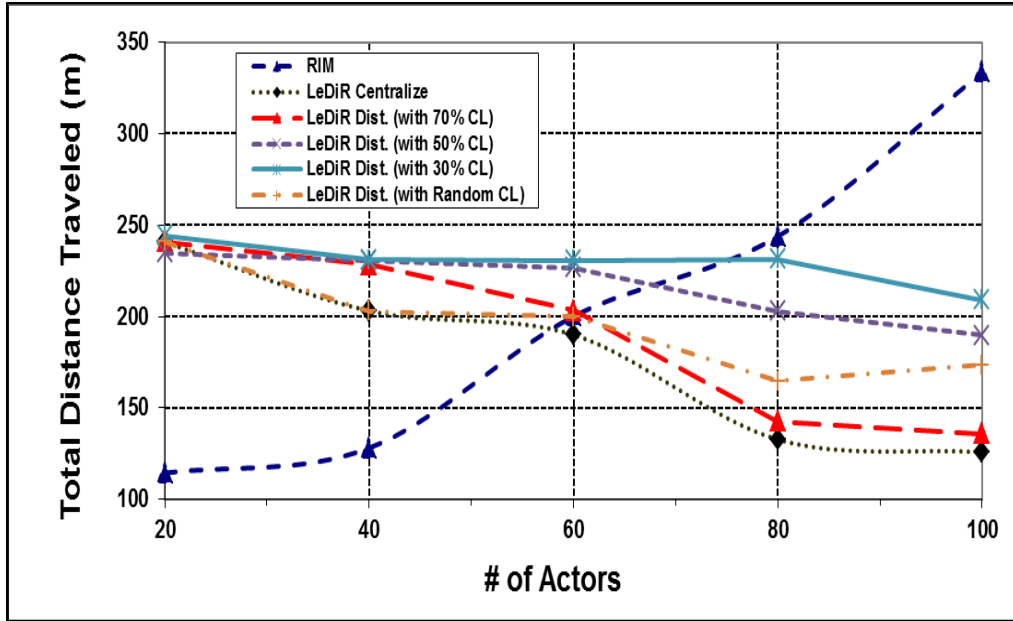
Theorem 2 proves that the maximum number of nodes involved in the recovery process is $O(N)$. In addition, Theorem 5 proves that the maximum distance a node travels in the recovery process is r . Suppose t is the time to travel distance r . Assuming the worst case scenario where the node movement is sequential, the total time to restore network connectivity would be $(N) * t$. Thus, the maximum convergence time of LeDiR to restore inter-actor connectivity is $s + (N) * t$ which is $O(N[d^2+t])$. For a uniform actor distribution, the value of d depends only on r [40]. Thus, both d and t can be considered constant and the inter-actor connectivity would be restored in $O(N)$. □

5.7 Performance Evaluation of LeDiR

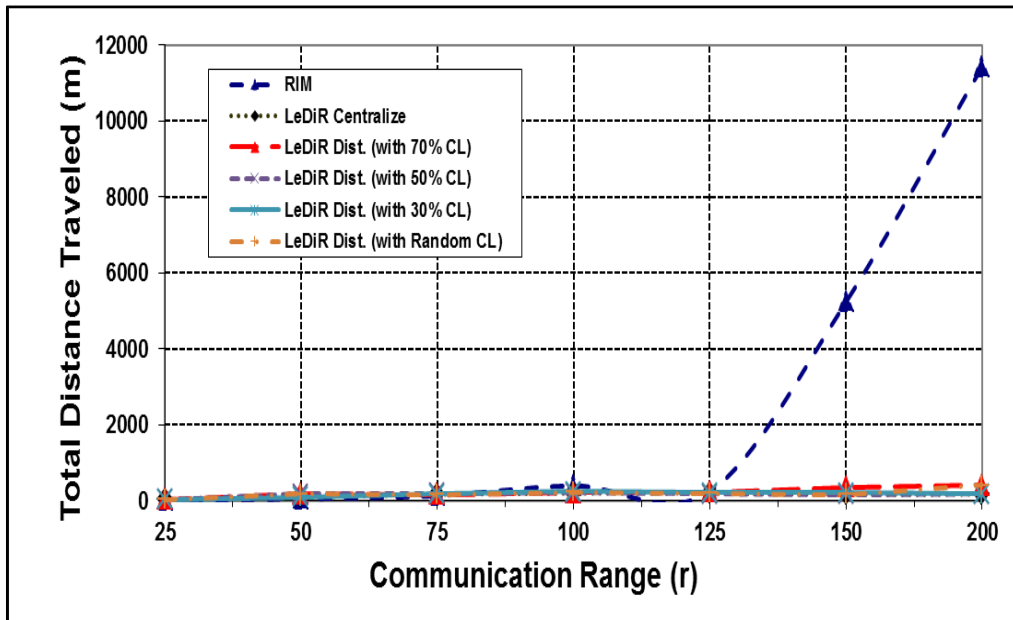
As we mentioned earlier that LeDiR strives to restore the network connectivity while minimizing the recovery overheads and maintaining the shortest path lengths at their pre-failure value. We group the results into two sets: (1) overhead related metrics and (2) path length validation metrics. We compare the performance of LeDiR to RIM [11] and DARA [10], which are the most effective published solutions for the tolerance of a single node failure in WSAN.

The first set compares LeDiR, which runs in a distributed manner, to a centralized version that provides the least traveled distance. We also compare LeDiR to RIM in terms of the recovery overhead. LeDiR selects the smallest partition and tries to maintain the existing communication links between nodes within the block that will perform the recovery. The movement technique and operation is closer to RIM; in other words, RIM can achieve the same objective while DARA cannot guarantee it. Therefore, in the first set we compare LeDiR with RIM and not DARA.

In addition to the centralized version of LeDiR and RIM, the second set of simulation experiments compares LeDiR to DARA. The reason is that both RIM and DARA are designed particularly to restore the network connectivity. However, RIM and DARA do not care whether a pre-failure shortest path gets extended or not. Therefore, the path length validation metrics would assess how frequent the shortest paths are affected by contemporary recovery schemes and assess the contribution of LeDiR to sustaining the pre-failure path lengths.



(a)



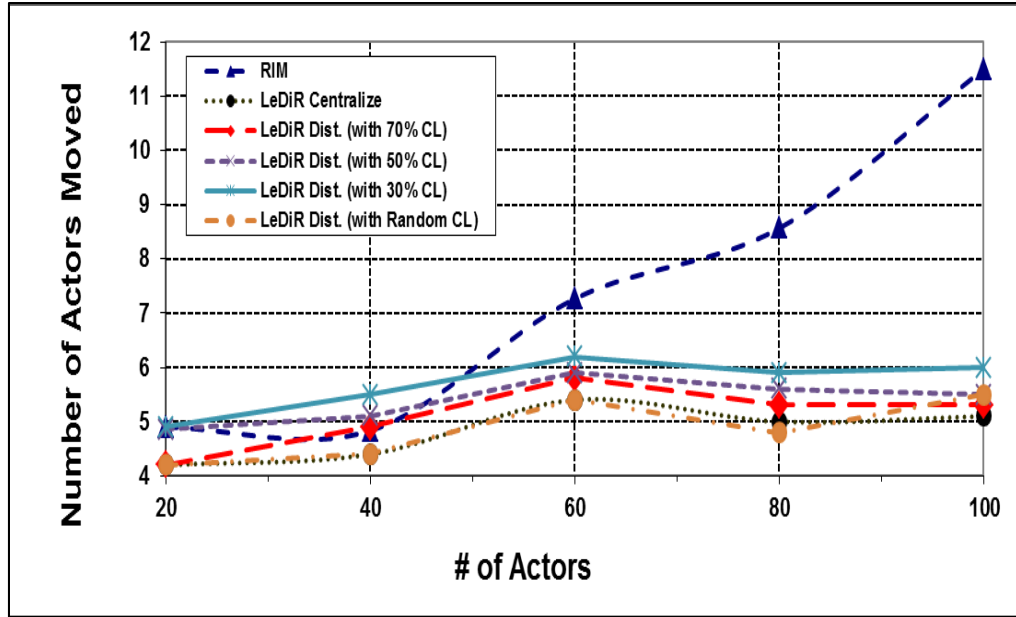
(b)

Figure 5.7: (a) Effect of the network size on the total distance traveled by actor nodes under RIM and LeDiR where CL is varying (with $r=100$); (b) The impact of an increased actor's communication range on the relocation overhead for a network of 100 actor nodes.

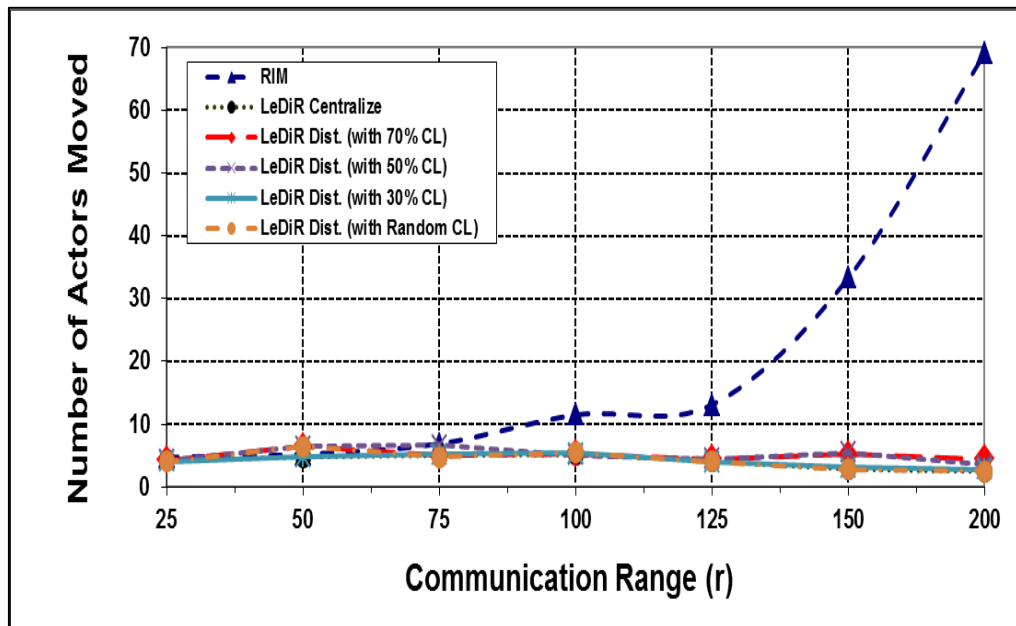
5.7.1 Overhead Related Metrics

Figure 5.7 shows the distance that actor nodes collectively travel during the recovery under varying r and N , respectively. Figure 5.7-(a) shows that LeDiR scales well with dense topologies and outperforms RIM significantly. Although in sparse topologies LeDiR does not appear to have advantage over RIM, RIM does not prevent the paths from being extended, as shown later in this section. More specifically, in networks with low degree of connectivity most nodes have few neighbors and RIM often yields a topology that has some longer paths between pairs of nodes compared to the pre-failure topology. When the node count increases, LeDiR demonstrates distinct performance and dominates RIM despite the path length constraint. Figure 5.7-(b) captures the impact of changing r for a network of 100 nodes. Obviously, LeDiR performs very well in highly connected networks and matches the performance of RIM for low ranges, yet meeting the inter-node path length goal.

As we mentioned earlier, the decrease in the CL level means fewer entries in the actor's SRT and less information for actor to make the right assessment of the scope of the failure and define the most appropriate recovery plan. This leads to an increase in the likelihood of wrong decision making and results in more travel overhead. However, Figure 5.7 shows LeDiR stays robust and yields close to optimal results when CL is 70%. Even under very low CL scenarios, e.g. 30%, the performance of LeDiR is not far from the centralized version.



(a)



(b)

Figure 5.8: Number of actors that moved during the recovery while varying (a) the communication range (with $N = 100$), and (b) the network size (with $r = 100$).

While simulating LeDiR, initially we assume that all the nodes are deployed together and thus have almost same confidence level (CL). We have further tested the performance of LeDiR with heterogeneous CL, i.e. some nodes with 30%, some with 50%, some with 70%. This mimics the case when nodes are deployed in batches and the case when the traffic density is different throughout the network. In Figure 5.7-(a) and (b), the curve for LeDiR with Random CL reflected the performance with heterogeneous CL values and the results are very close to those of a centralized implementation of LeDiR.

Figure 5.8 indicates clearly that LeDiR outperforms RIM by moving fewer nodes during the recovery, especially for dense and highly connected topologies. Unlike RIM, LeDiR try to relocate nodes that belong to the smallest block in order to avoid triggering large scale movement of child actors. In addition, networks with high node density or large radio range are highly connected; thus cut-vertices usually exist close to the network periphery. Determining the smallest block would then limit the scope of the recovery and make LeDiR more advantageous.

With respect to the number of messages, LeDiR introduces significantly less messaging overhead in comparison to the centralized version and RIM as shown in Tables 5.2 and 5.3. Actually, in the centralized version, each node must be aware of the complete network topology. Thus, the messaging overhead grows dramatically as the nodes count increases. On the other hand, RIM requires maintaining 1-hop neighbor information. Conversely, LeDiR leverages the available route discovery process and does not impose pre-failure messaging overhead. The only communication cost incurred during the

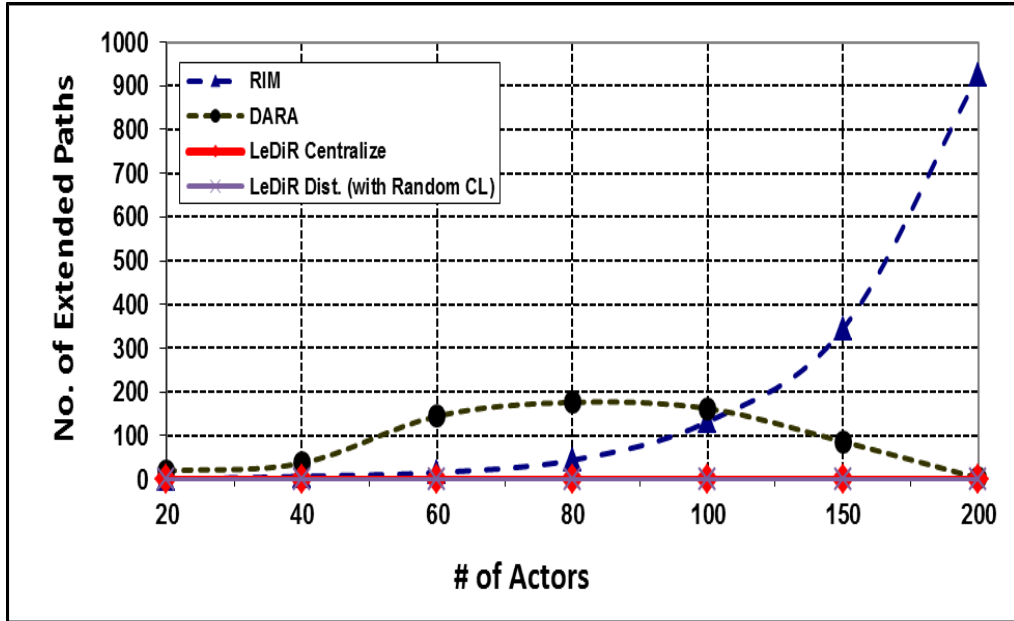
recovery is when a node informs its children about its movement or broadcasts the successful relocation.

TABLE 5.2: # of Messages Sent by LeDiR with varying # of actors

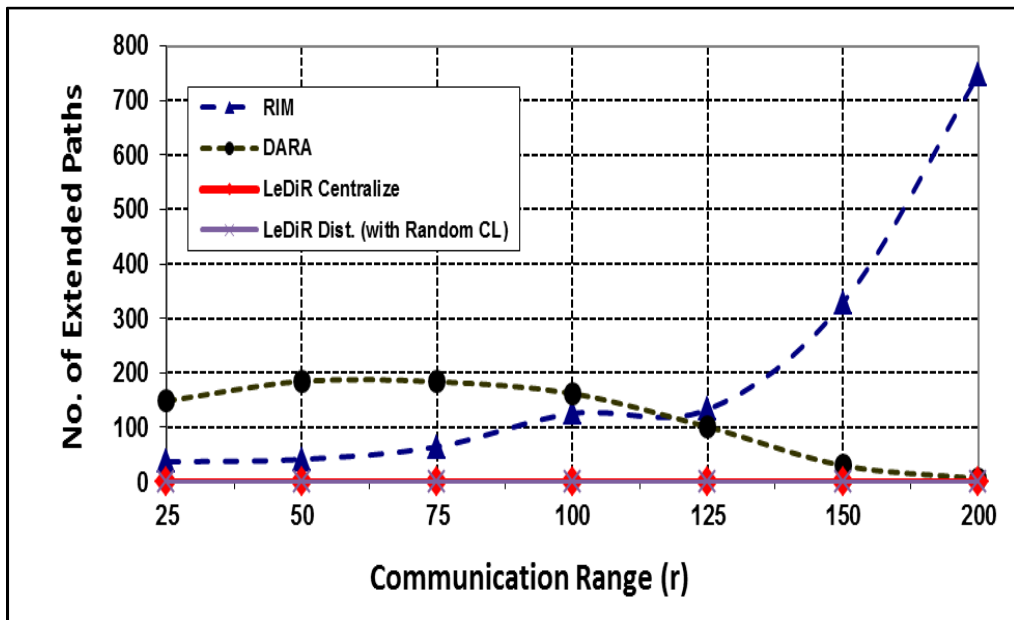
# of Actors	RIM	LeDiR				
		Centralized	Distributed			
			70% CL	50% CL	30% CL	Rand. CL
20	30	406	6	6	9	7
40	56	1608	9	10	14	12
60	85	3613	13	15	17	14
80	115	6406	11	19	25	17
100	152	10017	17	20	30	23

TABLE 5.3: # of Messages Sent by LeDiR with varying actor radio range

Radio Range	RIM	LeDiR				
		Centralized	Distributed			
			70% CL	50% CL	30% CL	Rand. CL
25	112	10010.9	11	11	13	12
50	113	10009.4	14	14	9	16
75	121	10010.7	21	15	24	17
100	163	10017.3	17	23	26	22
125	143	10013.4	25	25	23	26
150	351	10016.2	71	29	87	33
200	1072	10021.1	78	56	98	62



(a)



(b)

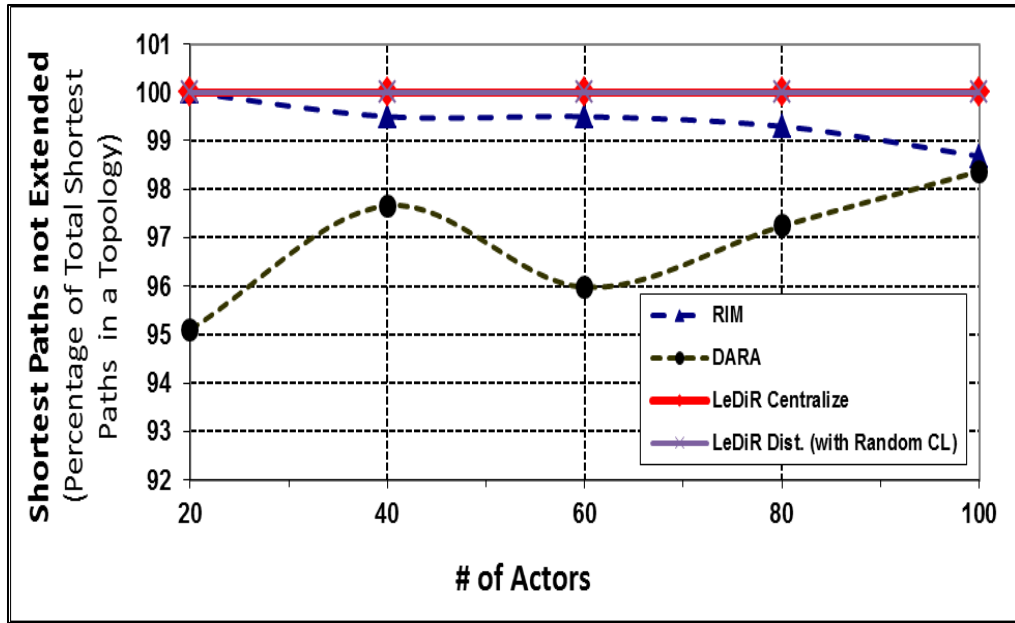
Figure 5.9: The number of extended paths per topology [average over 30 runs] after performing the recovery while varying (a) the communication range (with $N=100$), and (b) the network size (with $r=100$).

5.7.2 Path Length Validation Metrics

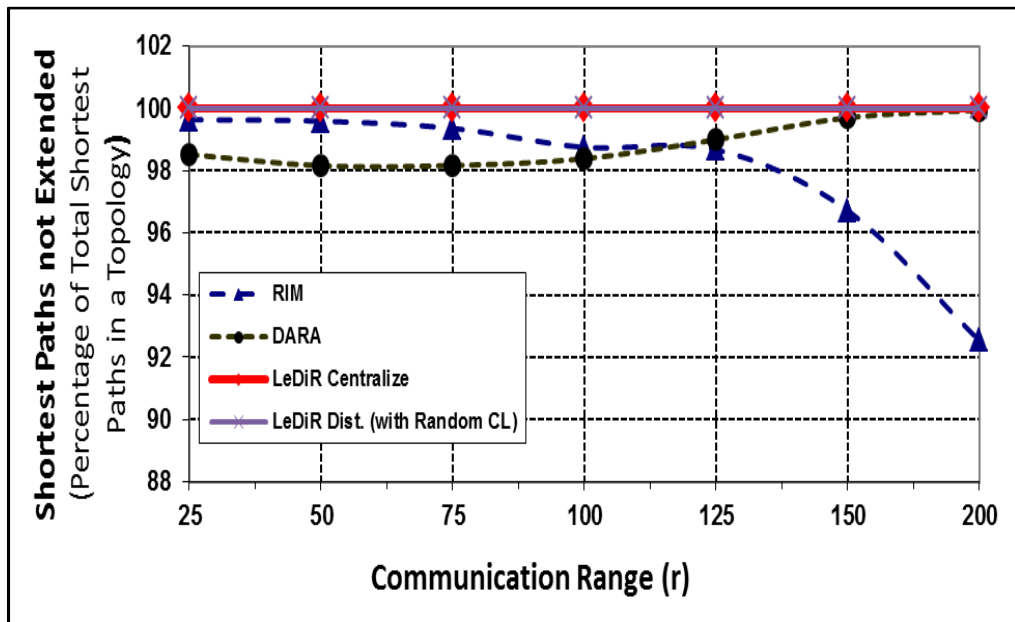
Figure 5.9 validates that LeDiR does not extend the shortest path between any pair of nodes. We compare LeDiR to RIM and DARA. As expected, LeDiR achieves its design objective and does not extend any shortest path unlike RIM and DARA. RIM engages all neighbors of the failed node and triggers subsequent cascaded relocation. This can be tolerated in sparse topologies. However in highly connected networks, i.e. large N or r values, many nodes are involved in the recovery process as indicated by Figure 5.8. As a result the scope of node movement grows dramatically and the number of extended paths increases as shown in Figure 5.9.

On the other hand DARA performs very close to LeDiR in highly connected topologies. In sparse networks DARA does not do well with significant number of extended paths. However, after a certain point the number of extended paths in the network started to decline. In Figure 5.9-(a) this started to happen when the number of actors reaches 80 and in Figure 5.9-(b) when the actor's radio range exceeds 75 meters. The reason is that in highly connected topologies, cut-vertices are found only at or near the network periphery. This particularly is very advantageous for DARA since the network would be partitioned into a very large block and few small blocks. In such a case, DARA would select the best candidate node, i.e., that with least node degree, from a small block since the large block would be highly connected.

DARA performs significantly worse than LeDiR in sparse topologies. This is attributed to the fact that DARA selects the neighbor with the least degree to replace the faulty



(a)



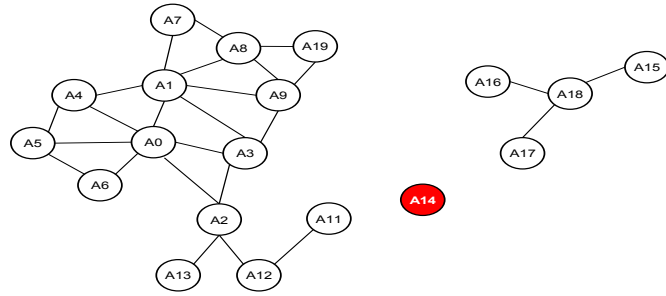
(b)

Figure 5.10: Percentage of shortest paths that are NOT extended per topology during the network recovery [average over 30 runs] while varying (a) the communication range (with $N=100$), and (b) the network size (with $r=100$).

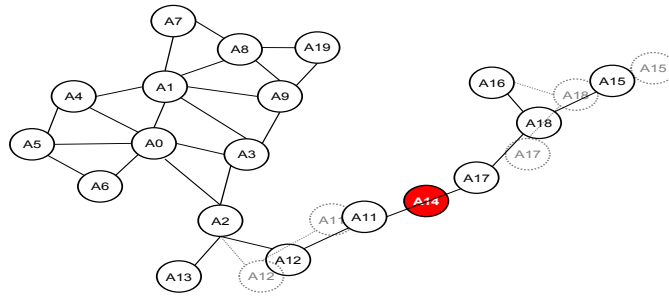
actor. DARA does not care whether the selected node belongs to the smallest block. As a result, a node from a significantly large block may move to replace the faulty node. This cause many cascaded movements that extend many of the shortest paths between nodes. The graphs in Figure 5.10-(a) and (b) show the percentage of total number of shortest paths in a topology that do not get extended. Clearly, for LeDiR, the curve stays at 100%. DARA improves when adding more nodes or increasing the radio range since the network connectivity grows. However, RIM performs very close to LeDiR with sparse network and pretty poor with densely populated topologies, as also noted in Figure 5.9.

Figure 5.11 illustrates the difference between LeDiR and the baseline approaches when the node density around the failed node is low. To restore connectivity after the failure of actor A_{14} , LeDiR would move actor A_{17} from the smallest block. RIM would stretch the links from both disjoint blocks and move A_{11} and A_{17} toward A_{14} , to reconnect the network. The cascaded relocation for either LeDiR or RIM would not increase any shortest path. However, when applying DARA, actor A_{11} will move to replace A_{14} . Actors A_{12} , A_2 and A_{13} will move during the cascaded relocation. Thus, as a result many shortest paths got increased, e.g. the path from A_3 to A_{17} , causing DARA to perform poorly compared to LeDiR or RIM.

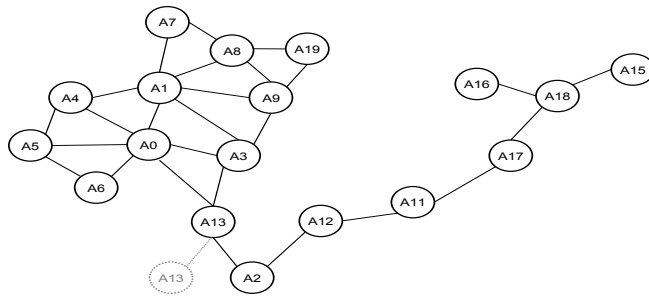
Another very important question is whether increasing the size of the network has any effect on the level of path growth in the repaired topology. For example, if we could run DARA for a network of 200 actors, would that lengthens some paths by 3 hops or more? In our simulations experiments, we have observed that the extension in the path length is



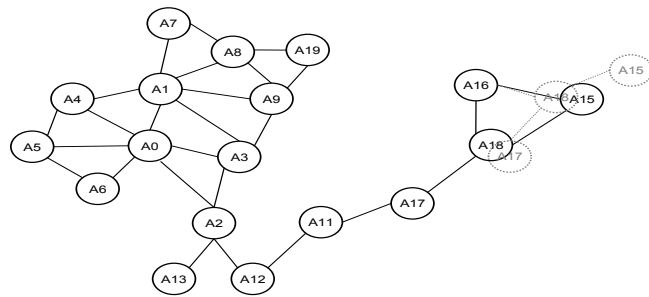
(a)



(b)



(c)



(d)

Figure 5.11: (a) WSN with a sparse 1-connected topology and a faulty node. (b) Topology recovered by using RIM (c) topology recovered by using DARA and (d) topology recovered by using LeDiR .

independent of the number of nodes in the network and is entirely dependent on which node was selected to replace the faulty actor. For example, if node F fails and A , B and C are the 1-hop neighbors. One of them should replace F to re-establish the network connectivity. Now suppose C is also a leaf node, thus moving C to replace F would not increase any shortest path length and DARA would act like LeDiR no matter what the network size is.

Basically in DARA, the length of the shortest path may grow because a parent moves to replace a faulty node and a child node is inserted in the path to bridge the gap that was created due to the parent departure. In addition, the cascaded nature of movement in DARA and RIM also plays a role since it may get a node to depart one shortest path and join another during the recovery operation. Thus, it is obvious that more paths get extended when the network grows. However, the increase in terms of number of hops would remain low, i.e., 1 or 2 in many cases. The increase in path length boosts the potential of packet loss and data delivery delay, and negatively impacts the application. LeDiR strives to avoid that.

5.7.3 General Comments

We would like to make few additional notes about the performance and applicability of LeDiR. First, LeDiR is designed to recover from a single node failure. Simultaneous node failure may cause conflicting conditions for LeDiR to converge successfully. As mentioned earlier, the probability for multiple nodes to fail at the same time is very small and would not be a concern for LeDiR. Second, LeDiR tends to shrink the smallest block

inward towards the failed node, it may negatively affect the node coverage. In general, the impact on coverage would depend on the relationship between the radio and sensing ranges. One would argue that unless the coverage range is significantly larger than the radio range, the loss of a node will have more dominant impact on the coverage than the connectivity restoration process. The focus of the LeDiR approach is on connectivity and does not factor in the impact on coverage. We plan to consider a joint connectivity and coverage recovery metric in the future.

5.8 Concluding Remarks on LeDiR

This chapter discusses an important problem in mission critical WSANs; that is re-establishing network connectivity after node failure without extending the length of data paths. We have proposed a new distributed Least-Disruptive topology Repair (LeDiR) algorithm that restores connectivity by careful repositioning of nodes. LeDiR relies only on the local view of the network and does not impose pre-failure overhead. LeDiR can recover from a single node failure at a time. Generally, simultaneous node failures are very improbable unless a part of the deployment area becomes subject to a major hazardous event, e.g., hit by a bomb. The performance of LeDiR has been validated through rigorous analysis and extensive simulation experiments. The experiments have also compared LeDiR to a centralized version and to contemporary solutions in the literature. The results have demonstrated that LeDiR is almost insensitive to the variation in the communication range. LeDiR also works very well in dense networks and yields close to optimal performance even when nodes are partially aware of the network topology.

CHAPTER 6

RESTORING CONNECTIVITY WITH MINIMAL NODE MOVEMENT

6.1 Introduction

Most of recovery schemes which required just 1-hop neighbor information are not efficient since they often involve many actors and require long travel distances. This chapter discusses LeMoToR which actually is an extension of LeDiR. Like LeDiR, LeMoToR utilizes existing path discovery activities to get and maintain topology related information and imposes no additional pre-failure communication overhead. In LeDiR, the routing cost is not counted towards communication overhead of the proposed algorithm since data has to be routed anyway regardless the proposed algorithm is applied or not. This is valid for LeMoToR as well. Like LeDiR, LeMoToR relies on the local view of a node about the network to orchestrate an autonomous restoration of the strong connectivity. On the other hand, LeMoToR does not impose the constraint to sustain the path length between any pair (i, j) of node at pre-failure status. The objectives

are to minimize the node relocation, total travel distance and communication overhead. Like LeDiR, LeMoToR opts to localize the recovery process and operates in a distributed manner. When a node fails, its neighbors will individually consult their possibly-incomplete routing table to decide on the appropriate course of action and define their role in the recovery if any. If the failed node is a cut-vertex, i.e., a node that causes the network to partition into disjoint blocks, the neighbor node that belongs to the smallest block reacts. However, the main difference is that unlike LeDiR, LeMoToR is applied recursively to sustain the intra-smallest-block connectivity. When a node moves, its neighbors repeat the LeMoToR connectivity restoration process. In brief, the goal of LeMoToR is to reconnect the disjointed network while keeping the node movement as minimum as possible and involving the least number of actor nodes in the recovery process.

In the subsequent section, we give an overview of LeMoToR as a centralized solution. Section 6.3 provides a detail example on recovery with minimal node movement. Distributed implementation of LeMoToR is explained in section 6.4 and section 6.5 explains the pseudo code. Section 6.6 provides discussion on performance evaluation of LeMoToR. Concluding remarks on LeMoToR are presented in section 6.7.

6.2 LeMoToR – Main Steps

The following subsections highlight the major steps in LeMoToR.

6.2.1 Failure Detection

To detect a node failure in the neighborhood, an exchange of heartbeat messages is assumed in the network. After n missing heartbeats, a node F would be assumed faulty. If the failed node F is a cut-vertex, network recovery measures would be triggered on the 1-hop neighbors of F . Similar to LeDiR, cut-vertex (critical node) detection is done by using the SRT.

6.2.2 Smallest Block Identification

As mentioned earlier, after a cut-vertex (critical node) failure the 1-connected network G is split into more than one connected component, i.e., sub-network $sub(G)$. Each $sub(G)$ consists of few nodes of G that are 1-connected to each other within the $sub(G)$. Basically, each $sub(G)$ is a separate “*block*” that was connected to the other blocks in G via faulty cut-vertex. LeMoToR attempts to find a block among the disjoint blocks that consists of the least number of nodes, referred to hereafter as the “*smallest block*”. Actually, LeMoToR aims to confine the node movement within the smallest block to minimize the node movement. To identify the smallest block, every 1-hop neighbor of faulty node would identify the reachable set of nodes for itself and every other 1-hop neighbor of the failed node by using SRT. The block with the fewest nodes is identified as a smallest block.

6.2.3 Replacing the Faulty Node

To replace the faulty node F , a neighbor node J is selected from the smallest block. The reason is that LeMoToR strives to minimize the number of node movements during the

network recovery. Since LeMoToR is recursive in nature, moving a node and its children from the smallest block would most probably involve the fewest actor nodes in the recovery. In case more than one actor with such characteristics exists, the closest actor to the faulty node would be picked. Any further ties will be resolved by selecting the actor with the least ID.

6.2.4 Children Movement

When node J moves to replace the faulty node, possibly some of its children will become disconnected. To regain the connectivity, children would assume the moved parent node as a dead node and would apply LeMoToR at the children level. The smallest block at the children level would be identified. The child that belongs to the smallest block would proceed to the location of already moved parent node. This phenomenon would continue until all the nodes are reconnected with the network G .

6.3 Recovery with Minimal Node Movement: An Example

Figure 6.1 shows an example for how LeMoToR restores connectivity after the failure of A_{10} . Obviously, node A_{10} is a cut-vertex and A_{14} becomes the 1-hop neighbor that belongs to the smallest block (Figure 6.1-(a)-(c)). In Figure 6.1-(d), node A_{14} notifies its neighbors and moves to the position of A_{10} to restore connectivity. Disconnected children, nodes A_{15} and A_{16} , execute LeMoToR again to find out which one of them should move to the location of A_{14} . Obviously, node A_{15} belongs to the smallest block and thus moves to the location of A_{14} to maintain the communication link (Figure 6.1-(e)). Note that the reason to execute LeMoToR recursively and to identify the smallest block even among

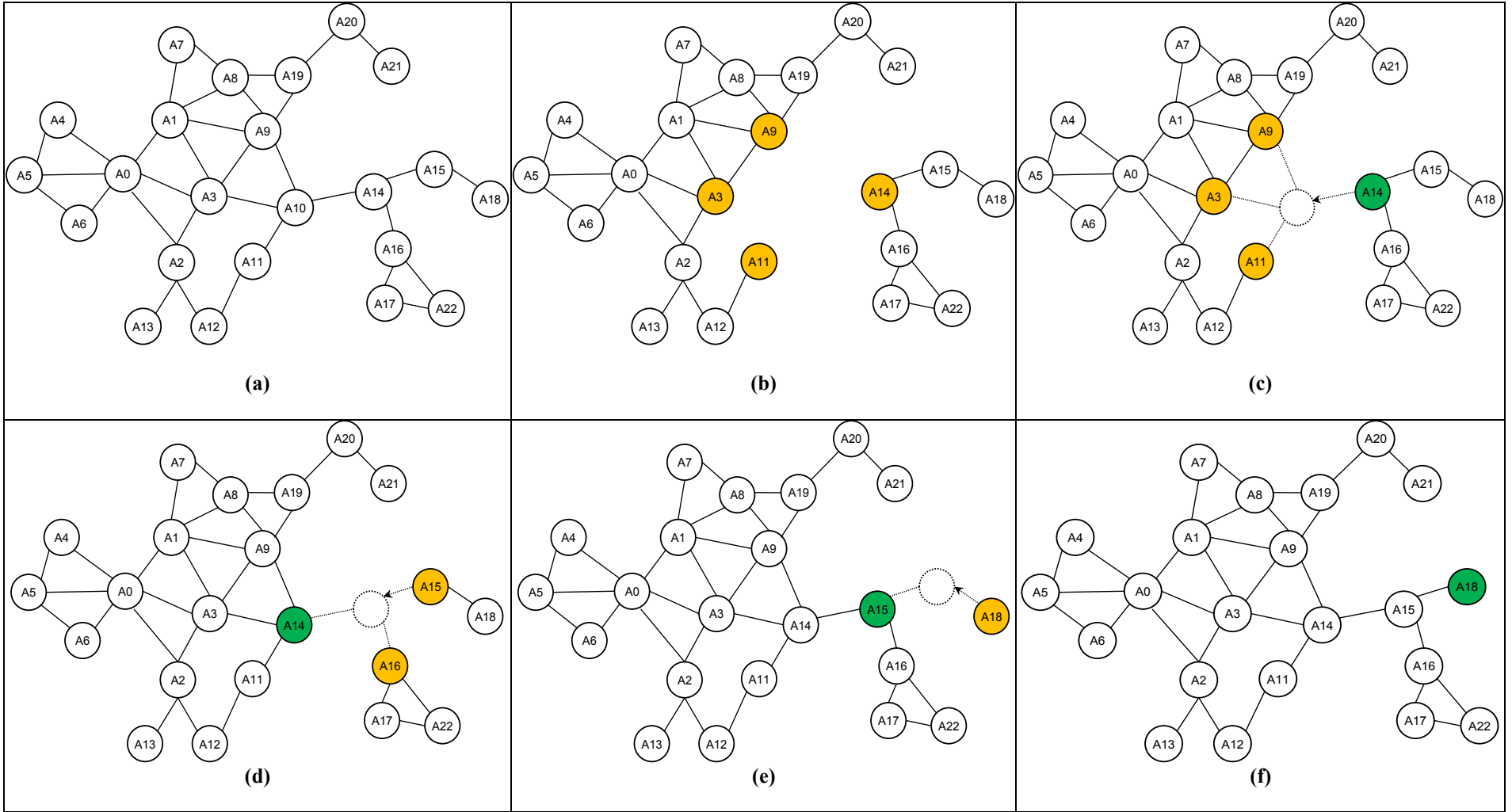


Figure 6.1: An example illustrating how LeMoToR restores connectivity after the failure of node A_{10} .

the children movement is to minimize the overall node movement. Nodes A_{15} would notify its only child A_{18} , before it moves. Since A_{18} is the only child, it simply belongs to the smallest block and moves to location of A_{15} (Figure 6.1-(f)). Figure 6.1-(f) shows the repaired network.

6.4 Distributed LeMoToR Implementation

Distributed implementation of LeMoToR is very similar to LeDiR. The assumptions and issues for a distributed implementation of LeDiR stated under section 5.4 are applicable to LeMoToR as well. Similar to LeDiR, every node in the topology may potentially have different CL from others. In such a case, upon the detection of a node failure the neighboring nodes may have an inconsistent assessment of the impact of the node loss on the network and on which actor is the best candidate for leading the recovery. We argue nonetheless that this is rare in practice since the mobility pattern among actors is not typically high given their involvement in actuation activities. In addition, the operation in WSN is collaborative in nature and an actor usually communicates with many others and thus the routing table would not be sparse.

As in LeDiR, the second issue is determining of the best candidate, i.e., the neighbor of the failed node that belongs to the smallest block. In case, a neighbor of faulty node that belongs to the smallest block does not have sufficient entries in its SRT it would not know that it belongs to the smallest block and would not thus initiate the recovery process by moving to replace the faulty node. Since the neighbors of faulty node cannot reach each other, a partially populated SRT may lead to a deadlock with none of the

neighbors of faulty node responding to the failure and leaving the network disconnected. To handle this issue, LeMoToR imposes a time-out after which the neighbor(s) belonging to the second largest block will move. This time multiple neighbors may be potentially moving towards faulty node. To avoid having more than one actor replacing faulty node, LeMoToR requires these nodes to broadcast messages with their ID so that they pause as soon as reaching other neighbors of faulty node that happen to be in a different block. The pause time would allow these neighbors to negotiate and pick the best candidate to continue on to the position of faulty node.

6.5 LeMoToR Pseudo Code

Figure 6.2 shows the pseudo code of LeMoToR. A node J would trigger LeMoToR, whenever a cut-vertex node failure is detected in the 1-hop neighborhood (line 1-2). Node J would test its eligibility to move to replace the faulty node by executing the *IsBestCandidate()* procedure (line 3). Basically, the procedure *IsBestCandidate()* finds whether node J belongs to the smallest disjointed sub-network block. If so, node J notifies its children (line 4-10) and moves to the location of the faulty node. Otherwise, node J checks whether it is to perform a movement to sustain current communication links (line 11), and if so it executes LeMoToR (line 15) to find whether it belongs to the smallest block. If so, it moves to the location of the already moved parent node to maintain the communication link. Nodes only move once (line 12-14). LeMoToR would be executed on the children node that loses direct communication link to the moved parent (neighbor).

```

// Every node builds its shortest path routing table (SRT) based
// on the route discovery activities that it initiates or serve in, e.g.
// while executing a distributed routing protocol.

```

LeMoToR(*J*)

```

1 IF node J detects a failure of its neighbor F
2   IF neighbor F is a cut-vertex node
3     IF IsBestCandidate(J)
4       Notify_Children(J);
5       J moves to the Position of neighbor F;
6       Moved_Once ← TRUE;
7       Broadcast(Msg('RECOVERED'));
8       Exit;
9     END IF
10  END IF
11 ELSE IF J receives (a) notification message(s) from F
12   IF Moved_Once || Received Msg('RECOVERED')
13     Exit;
14   END IF
15   LeMoToR(J)
16 END IF

```

IsBestCandidate (*J*)

```

// Check whether J is the best candidate for tolerating the failure
17 NeighborList[] ← GetNeighbors (F) accessing the column F in SRT;
18 SmallestBlockSize ← Number of nodes in the network;
19 BestCandidate ← J;
20 FOR each node i in the NeighborList[]
    //Use the SRT after excluding the failed node to find the
    //set of reachable nodes;
21   Number of reachable nodes ← 0;
22   FOR each node k in SRT excluding i and F
23     Retrieve shortest path from i to k by using SRT;
24     IF the retrieved shortest path does not include node F
25       No. of reachable nodes ← No. of reachable nodes + 1;
26     END IF
27   END FOR
28   IF Number of reachable nodes < SmallestBlockSize
29     SmallestBlockSize ← Number of reachable nodes;
30     BestCandidate ← i;
31   END IF
32 END FOR
33 IF BestCandidate == J
34   Return TRUE;
35 ELSE
36   Return FALSE;
37 END IF

```

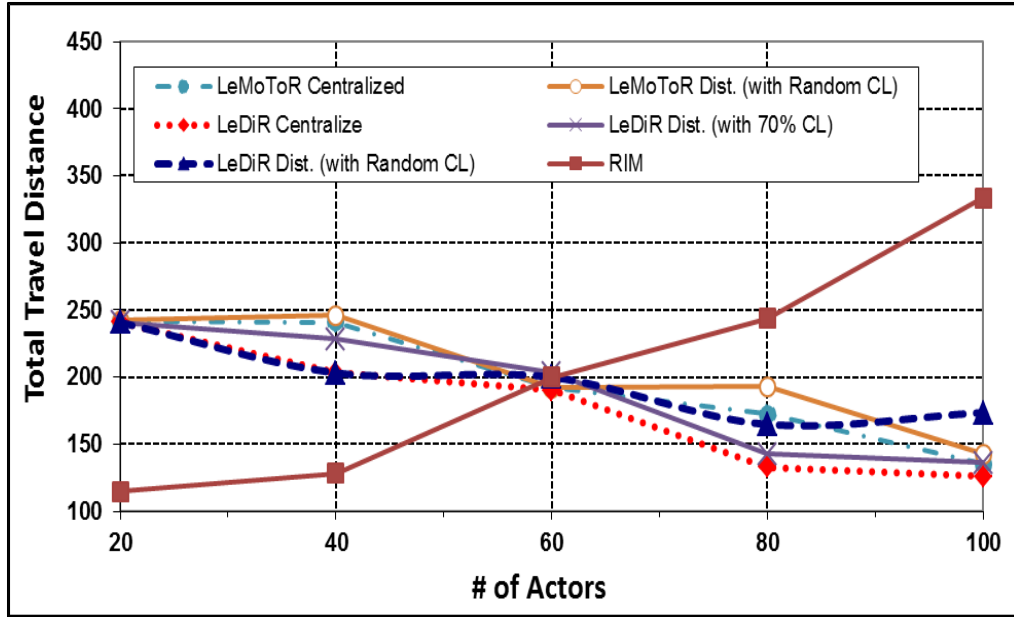
Figure 6.2: Pseudo code of LeMoToR

6.6 Performance Evaluation of LeMoToR

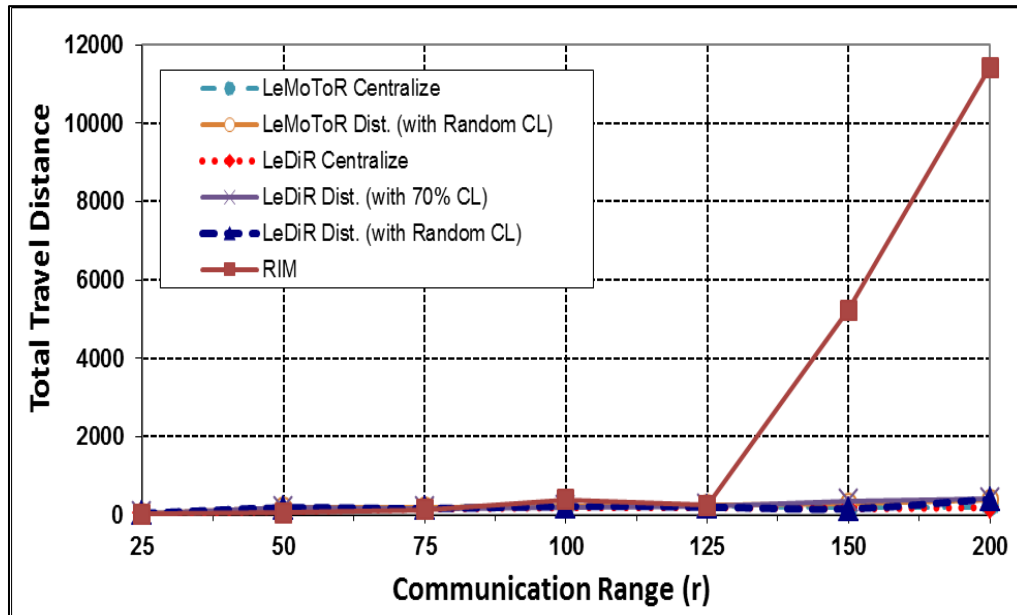
The performance of LeMoToR is validated through simulation. As we mentioned earlier LeMoToR strives to restore the network connectivity while minimizing the number of relocated nodes. We compare the performance of LeMoToR to LeDiR, and RIM [11]. The movement technique and operation of LeMoToR is closer to RIM than any other published scheme since it is designed particularly to restore the network connectivity with minimum messaging overhead. In addition, LeMoToR resembles LeDiR with the exception of the relaxation of the path-length constraint.

Figure 6.3 shows the total travelled distance overhead under all considered approaches. It clearly indicates that LeMoToR and LeDiR have nearly similar performance and scale well as the network gets larger. However, in sparse topologies LeMoToR does not appear to have advantage over RIM. While RIM has outperformed all other schemes for small network size, its performance degrades steadily as the network size grows. Considering the effect of communication range on the total travelled distance, Figure 6.3-(b) shows that LeMoToR has a very stable behavior and confirms the previous finding of minimum travelled distance. The efficiency of LeMoToR depends on the network traffic and activities since it directly affects how SRT is populated. Nonetheless, LeMoToR does still converge even if partial SRT is available.

As stated earlier, the decrease in the CL level means fewer entries in the actor's SRT and less information for actor to make the right assessment of the scope of the failure and define the most appropriate recovery plan. This leads to an increase in the likelihood of



(a)



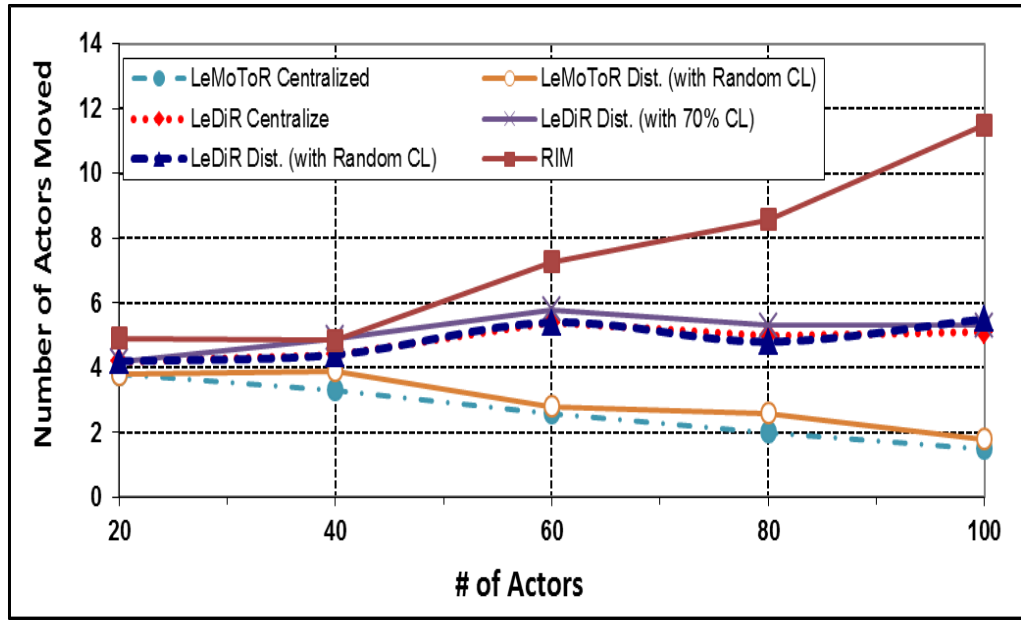
(b)

Figure 6.3: The total distance traveled by actor nodes where (a) network size is varied (with $r=100$), (b) communication range is varied (with $N=100$)

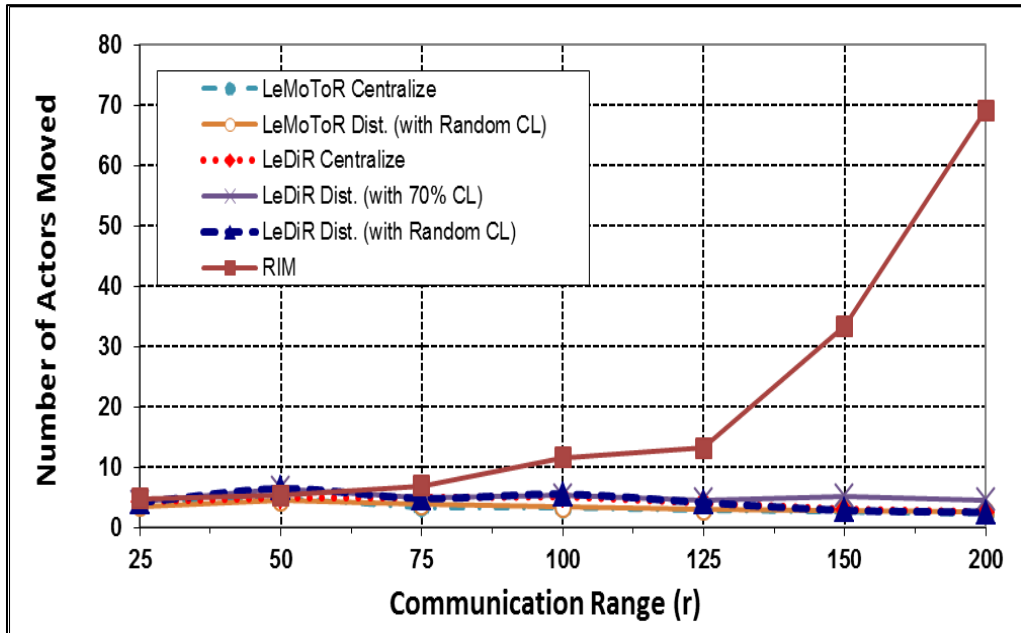
wrong decision making and results in more travel overhead. We noticed that this happens when the number of entries in the SRT is below 30% for all the nodes in the topology which is very rare. However, Figure 6.3 shows that LeMoToR stays robust and yields results close to the optimal with random CL. In other words, despite the incomplete SRT that some nodes have, i.e., LeMoToR's performance matches the centralized implementation that bases the decision on knowing the entire network topology.

While simulating LeMoToR, initially we assume that all the nodes are deployed together and thus have almost the same confidence level (CL). In other words, all nodes are placed in the topology with the same number of shortest path routing entries in their SRTs. Furthermore, we have tested the performance of LeMoToR with heterogeneous CL; means that in the same topology some nodes are missing 30% SRT entries, some missing 50% SRT entries, and some missing 70% SRT entries. This mimics the case when nodes are deployed in batches and the case when the traffic density is different throughout the network. In Figure 6.3-(a) and (b), the curves for LeMoToR and LeDiR with Random CL reflect the performance with heterogeneous CL values and the results are very close to those of centralized implementations.

Considering the number of relocated nodes during the recovery process, Figure 6.4 indicates clearly that LeMoToR outperforms all other approaches by moving fewer nodes during the recovery, especially for dense and highly connected topologies. Unlike RIM, LeMoToR and LeDiR try to relocate nodes that belong to the smallest block in order to avoid triggering large scale movement of child actors. Furthermore, LeMoToR extends



(a)



(b)

Figure 6.4: Number of actors that moved during the recovery while varying (a) the network size (with $r = 100$), (b) the communication range (with $N = 100$).

TABLE 6.1: # of Messages Sent by LeMoToR with varying # of actors

# of Actors	RIM	LeDiR		LeMoToR	
		Central.	Dist. Rand. CL	Central.	Dist. Rand. CL
20	30	406	7	402.8	4.6
40	56	1608	12	1604.2	6.2
60	85	3613	14	3605.75	5.2
80	115	6406	17	6401.1	5.8
100	152	10017	23	10010.5	6.1

TABLE 6.2: # of Messages Sent by LeMoToR with varying actor radio range

Radio Range	RIM	LeDiR		LeMoToR	
		Central.	Dist. Rand. CL	Central.	Dist. Rand. CL
25	112	10010.9	12	10005.4	8
50	113	10009.4	16	10004.6	10.4
75	121	10010.7	17	10007.4	10.6
100	163	10017.3	22	10011.8	16.7
125	143	10013.4	26	10010.2	12.8
150	351	10016.2	33	10015	14.4
200	1072	10021.1	62	10018.9	50.5

the application of this mechanism to all child actors. This feature makes LeMoToR relocates the least number of actor nodes among contemporary approaches.

With respect to the number of messages, again LeMoToR does very well by introducing noticeably less messaging overhead as shown in Tables 6.1 and 6.2. While RIM requires

maintaining 1-hop neighbor information, LeMoToR as well as LeDiR leverage the available route discovery process and do not impose pre-failure messaging overhead. The only communication cost incurred during the recovery is when a node informs its children about its movement or broadcasts the successful relocation. Nevertheless, LeMoToR requires fewer messages than LeDiR.

6.7 Concluding Remarks on LeMoToR

The collaborative and autonomous operation of the actors requires sustaining connectivity at all time and thus an actor failure must be tolerated in a distributed manner while imposing the least overhead. This chapter focuses on this important problem and proposed a new distributed Least-Movement Topology Repair (LeMoToR) algorithm. LeMoToR relies only on the local view of the network and does not impose pre-failure overhead. The performance of LeMoToR has been validated through extensive simulation experiments. We have also compared LeMoToR to a centralized version and to two other published schemes. The results have demonstrated that LeMoToR relocates the least number of actors to reestablish network connectivity after failure. LeMoToR also works very well in dense networks and matches the performance of the centralized implementation despite the partial knowledge that the nodes have about the network topology.

CHAPTER 7

CONCLUSION & FUTURE WORK

This chapter summarizes the thesis work and its contributions. In our research, we have tackled an important problem in mission critical WSANs; that is re-establishing movement-assisted and application-aware network connectivity after a critical node failure in WSANs. WSANs can serve applications in harsh environments, in which actor nodes may be subject to damage. The collaborative and autonomous operation of the actors requires sustaining connectivity at all time and thus an actor failure must be tolerated in a distributed manner while imposing the least overhead.

7.1 Conclusion

In this thesis, we propose three new distributed algorithms to restore movement-assisted and application-aware inter-actor network connectivity in WSANs. Firstly, we propose C^2AM to tackle the problem of connectivity repair after a node failure. C^2AM is a cross layer (network and application layers) approach that considers continuous sustenance of network connectivity and minimum application level disturbance. The experimental results demonstrate that C^2AM meets both goals of minimizing the actor travel distance

and communication overhead and maintaining application-level goals in a localized manner.

Secondly, we proposed a new distributed Least-Disruptive topology Repair (LeDiR) algorithm to re-establish network connectivity after a cut-vertex node failure without extending the length of data paths. LeDiR does not impose pre-failure overhead and relies only on the local view of the network. The performance of LeDiR has been validated through rigorous analysis and extensive simulation experiments. The experiments compare LeDiR to a centralized version and to contemporary solutions in the literature. LeDiR works very well in dense networks and yields close to optimal performance even when nodes are partially aware of the network topology.

Thirdly, we extend the LeDiR algorithm and call it Least-Movement Topology Repair (LeMoToR) to relocate the least number of nodes and reduce the traveled distance and message complexity. We compare LeMoToR to a centralized version and to two other published schemes. The results demonstrate that LeMoToR relocates the least number of actors to reestablish network connectivity after failure. In dense topologies, LeMoToR works very well and matches the performance of the centralized implementation.

7.2 Publications

It is worth to mention that all of three algorithms presented in this thesis have already been published in different well-known international IEEE and ACM conferences. In addition, we enhanced LeDiR further and a journal version has already been submitted to

IEEE Transection on Parallel and Distribute System. The citations of our published work are provided in the following:

1. Ameer Abbasi, U. Baroudi, M. Younis, K. Akkaya, "C2AM: An Algorithm for Application-Aware Movement-Assisted Recovery in Wireless Sensor and Actor Networks", in the Proceedings of 5th IEEE International Wireless Communications and Mobile Computing Conference (IWCMC-2009), Leipzig, Germany, June 2009.
2. A. Abbasi, M. Younis, and U. Baroudi, "Restoring connectivity in wireless sensor-actor networks with minimal topology changes," in the Proceedings of 2010 IEEE International Conference on Communications (ICC), Cape Town, South Africa, May 2010.
3. A. Abbasi, M. Younis, and U. Baroudi, "Recovering from a Node Failure in Wireless Sensor-Actor Networks with Minimal Topology Changes", submitted to IEEE Transactions on Parallel and Distributed Systems.
4. Ameer Abbasi, M. Younis, U. Baroudi, " Restoring Connectivity in Wireless Sensor-Actor Networks with Minimal Node Movement", in the Proceedings of 7th IEEE International Wireless Communications and Mobile Computing Conference (IWCMC-2011), Istanbul, Turkey, July 2011.

7.3 Future Work

After a critical node failure i.e. cut-vertex, movement-assisted network connectivity restoration is an important problem in mission critical WSANs especially when allocation-level constraints are also applicable on actors' mobility. In this thesis we proposed three new distributed algorithms to tackle such important issue in WSANs. However, all these three algorithms deal with the connectivity problems that occurred due to a single node failure and that can be handled locally by the 1-hop neighbors of the failed node.

Considering a problem with multiple/concurrent node failure is more complex and challenging in nature. Multiple/concurrent node failures can disjoint the network into multiple blocks that cannot be handled locally. In addition, failed nodes may be adjacent to each other in a single block or belong to different adjacent blocks. Therefore, the proposed algorithms cannot be applied directly to this complex problem.

To tackle such important and complex problem, initially, we plan to develop a centralized solution to handle multiple/concurrent node failures. This would also provide a baseline approach to validate distributed heuristics in the future.

In addition, terrain obstacle and localization error are not considered in this work. We simply assume that actor nodes can move straight to any desired location. In future, we plan to consider such issues for actor movement.

APPENDIX A

In this appendix A, we provide statistical analysis of our proposed algorithms. In our experiments, for each simulation setup 30 different network topologies are considered and the average values are reported. We observed that with 90% confidence level, the simulation results stay within 6% - 10% of the sample mean.

As we know that to get an impression of the expectation μ , it is sufficient to give an estimate. The appropriate estimator is the sample mean [43]:

$$\mu = \bar{X} = \frac{1}{n} \sum_{i=1}^n X_i$$

We take

$$1 - \alpha = 0.90$$

Assuming our samples follow normal distribution $N(\mu, \sigma / \sqrt{n})$ and Z is normalized normal distribution $N(0, 1)$, and then we have

$$P(-z_{1-\alpha/2} \leq Z \leq z_{1-\alpha/2}) = 1 - \alpha = 0.90$$

Where σ is standard deviation of our sample population and n is sample population which is 30 in our case. Then,

$$z_{1-\alpha/2} = 1.645$$

Thus, we get

$$\text{Confidence Interval}(CI) = \bar{X} \pm z_{1-\alpha/2} \frac{\sigma}{\sqrt{n}}$$

This might be interpreted as that with probability 0.90 we will find a confidence interval in which we will meet the parameter μ between the stochastic endpoints

$$\bar{X} - z \frac{\sigma}{\sqrt{n}} \quad \text{and} \quad \bar{X} + z \frac{\sigma}{\sqrt{n}}$$

TABLE A.1: Statistical Analysis of C²AM

# of Actors	Sample Mean (μ)	Standard Deviation (σ)	CI	% of Sample Mean
20	232.66923	47.44544	24.68085	10.60770
40	300.24465	57.45613	29.88837	9.95467
60	229.67860	29.45673	15.32322	6.67159
80	215.52396	39.28372	20.43518	9.48163
100	137.50551	22.23848	11.56833	8.41300

TABLE A.2: Statistical Analysis of LeDiR

# of Actors	Sample Mean (μ)	Standard Deviation (σ)	CI	% of Sample Mean
20	240.73833	35.45566	18.44384	7.66135
40	203.09998	40.34346	20.98645	10.33386
60	199.91418	35.43430	18.43273	9.22257
80	164.66023	23.34234	12.14256	7.37428
100	173.51990	32.24121	16.77170	9.66553

TABLE A.3: Statistical Analysis of LeMoToR

# of Actors	Sample Mean (μ)	Standard Deviation (σ)	CI	% of Sample Mean
20	240.73833	45.98579	23.92755	9.93612
40	203.09998	38.28778	19.91719	9.80616
60	199.91418	25.85762	13.45079	6.87349
80	164.66023	29.88876	15.54723	9.40094
100	173.51990	35.45222	18.44074	10.62032

REFERENCES

- [1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, “Wireless Sensor Networks: A Survey”, *Computer Networks*, Vol. 38, pp. 393-422, March 2002.
- [2] D. Estrin, R. Govindan, J. Heidemann, S. Kumar, “Next Century Challenges: Scalable Coordination in Sensor Networks”, in the *Proceedings of the 5th Annual International Conference on Mobile Computing and Networks (MobiCOM '99)*, Seattle, Washington, August 1999.
- [3] G. J. Pottie and W. J. Kaiser, “Wireless Integrated Network Sensors”, *Communications of the ACM*, Vol. 43, No 5, pp. 51 – 58, May 2000.
- [4] Katayoun Sohrabi, Jay Gao, Vishal Ailawadhi and Gregory J Pottie, “Protocols for Self-organization of a Wireless Sensor Network”, *IEEE Personal Communications*, Vol. 7, No. 5, pp. 16-27, October 2000.
- [5] R. Min, M. Bhardwaj, S. Cho, E. Shih, A. Sinha, A. Wang, A. Chandrakasan, “Low Power Wireless Sensor Networks”, in the *Proceedings of International Conference on VLSI Design*, Bangalore, India, January 2001.
- [6] J. M. Rabaey, M. J. Ammer, J. L. da Silva Jr. , D. Patel, S. Roundy, “PicoRadio Supports Ad Hoc Ultra-Low Power Wireless Networking”, *IEEE Computer*, Vol. 33, pp. 42-48, July 2000.
- [7] J.M. Kahn, R.H. Katz, K.S.J. Pister, “Mobile Networking for Smart Dust”, in the *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '99)*, Seattle, WA, August 1999.
- [8] M. Younis and K. Akkaya, “Strategies and Techniques for Node Placement in Wireless Sensor Networks: A Survey”, *The Journal of Ad-Hoc Networks*, Vol. 6, No.4, pp. 621-655, 2008.

- [9] I. F. Akyildiz, and I. H. Kasimoglu, “Wireless Sensor and actor networks: Research Challenges”, *Elsevier Ad hoc Network Journal*, Vol. 2, pp. 351-367, 2004.
- [10] Ameer Ahmed Abbasi, Mohamed Younis, Kemal Akkaya, “Movement-Assisted Connectivity Restoration in Wireless Sensor and Actor Networks”, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 20, No. 9, pp. 1366-1379, Sept. 2009, doi:10.1109/TPDS.2008.246
- [11] Mohamed Younis, Sookyoung Lee, Ameer Ahmed Abbasi, “A Localized Algorithm for Restoring Internode Connectivity in Networks of Moveable Sensors”, *IEEE Transactions on Computers*, Vol. 59, No. 12, pp. 1669-1682, Aug. 2010, doi:10.1109/TC.2010.174
- [12] Kemal Akkaya, Fatih Senel, Aravind Thimmapuram, Suleyman Uludag, “Distributed Recovery from Network Partitioning in Movable Sensor/Actor Networks via Controlled Mobility”, *IEEE Transactions on Computers*, Vol. 59, No. 2, pp. 258-271, Feb. 2010, doi:10.1109/TC.2009.120
- [13] A. Youssef, A. Agrawala and M. Younis, “Accurate Anchor-Free Localization in Wireless Sensor Networks”, in the *Proceedings of the 1st IEEE Workshop on Information Assurance in Wireless Sensor Networks (WSNIA 2005)*, Phoenix, Arizona, April 2005.
- [14] S. Yang, M. Li, and J. Wu, “Scan-Based Movement-Assisted Sensor Deployment Methods in Wireless Sensor Networks”, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 18, No. 8, pp. 1108-1121, August 2007.
- [15] G. Wang, G. Cao, and T. La Porta, “Movement-Assisted Sensor Deployment”, *IEEE Transactions on Mobile Computing*, Vol. 5, No. 6, pp. 640 -652, June 2006.
- [16] X. Li, H. Frey, N. Santoro, and I. Stojmenovic, “Localized Sensor Self-Deployment with Coverage Guarantee”, *ACM Sigmoblie Mobile Computing and Communications Review*, Vol. 12, No. 2, pp. 50-52), April 2008.

- [17] Y. Zou and K. Chakrabarty, "Sensor Deployment and Target Localization Based on Virtual Forces", in the *Proceeding of the 22nd Annual Joint Conference of the IEEE Computer and Communications (INFOCOM'03)*, San Francisco, CA, April 2003
- [18] G.T. Sibley, M.H. Rahimi, and G.S. Sukhatme, "Robomote: A Tiny Mobile Robot Platform for Large-Scale Sensor Networks", in the *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'02)*, Washington, DC, May 2002.
- [19] K. Dasgupta, M. Kukreja, K. Kalpakis, "Topology-Aware Placement and Role Assignment for Energy-Efficient Information Gathering in Sensor Networks", in the *Proceedings of the 8th IEEE Symposium on Computers and Communication (ISCC'03)*, Kemer-Antalya, Turkey, June 2003.
- [20] W. Youssef, M. Younis and K. Akkaya, "An Intelligent Safety-Aware Gateway Relocation Scheme for Wireless Sensor Networks", in the *Proceedings of the IEEE International Conference on Communications (ICC'06)*, Istanbul, Turkey, June 2006.
- [21] H. Liu, X. Chu, Y.-W. Leung, R. Du, "Simple Movement Control Algorithm for Bi-Connectivity in Robotic Sensor Networks", *IEEE Journal on Selected Areas in Communications*, Vol. 28, No. 7, pp 994-1005, August 2010.
- [22] K. Akkaya and M. Younis, "COLA: A Coverage and Latency aware Actor Placement for Wireless Sensor and Actor Networks", in the *Proceedings of the IEEE Vehicular Tech. Conf. (VTC-F'06)*, Montreal, Canada, Sept 2006.
- [23] G. Tan, S. A. Jarvis, and A.-M. Kermarrec, "Connectivity-Guaranteed and Obstacle-Adaptive Deployment Schemes for Mobile Sensor Networks", in the *Proceedings of the 28th International Conference on Distributed Computing Systems (ICDCS'08)*, Beijing, China, June 2008.
- [24] K. Sreenath, V. Giordano, and F. Lewis, "Avoiding Shared Resource Conflicts in Mobile Sensor Networks with Multiple Missions", *IET Control Theory & Applications*, Vol. 1, No. 3, pp. 665 – 674, May 2007.

- [25] G. Srivastava, P. Boustead, J. F. Chicharo, C. Ware, "TCMCA: A Source-Based Distributed Topology Control Algorithm for Mission Critical Applications in Mobile Ad-Hoc Networks", in the *Proceedings of the 11th IEEE International Conference on Networks, (ICON2003)*, Sydney, Australia, Oct. 2003.
- [26] P. Basu and J. Redi, "Movement Control Algorithms for Realization of Fault-Tolerant Ad Hoc Robot Networks", *IEEE Networks*, Vol. 18, No. 4, pp. 36-44, August 2004.
- [27] K. Akkaya, A. Thimmapuram, F. Senel, S. Uludag, "Distributed Recovery of Actor Failures in Wireless Sensor and Actor Networks", in the *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC 2008)*, Las Vegas, NV, March 2008.M.
- [28] Younis, Sookyoung Lee, Sheetal Gupta and Kevin Fisher, "A Localized Self-healing Algorithm for Networks of Moveable Sensor Nodes", in the *Proceedings of the IEEE GlobeCom, 2008*.
- [29] A. Alfadhly, U. Baroudi, M. Younis, "Optimal Node Repositioning for Tolerating Node Failure in Wireless Sensor Actor Networks", in the *Proceedings of the 25th IEEE Queen's Biennial Symposium on Communications (QBSC 2010)*, Kingston, Canada, June 2010.
- [30] S. Das, H. Liu, A. Kamath, A. Nayak, Ivan Stojmenović, "Localized Movement Control for Fault Tolerance of Mobile Robot Networks", in the *Proceedings of the 1st IFIP International Conference on Wireless Sensor and Actor Networks (WSAN 2007)*, Albacete, Spain, September 2007.
- [31] S. Das, H. Liu, A. Nayak, and I. Stojmenovic, "A Localized Algorithm for Bi-Connectivity of Connected Mobile Robots", *Telecommunication Systems.*, Vol. 40, No. 3-4, pp. 129-140, April 2009.
- [32] N. Tamboli and M. Younis, "Coverage-Aware Connectivity Restoration in Mobile Sensor Networks", in the *Proceedings of the IEEE International Conference on Communications (ICC 2009)*, Dresden, Germany, June 2009.

- [33] F. Senel, M. Younis, and K. Akkaya, "A Robust Relay Node Placement Heuristic for Structurally Damaged Wireless Sensor Networks", in the *Proceedings of the 34th IEEE Conference on Local Computer Networks (LCN 2009)*, Zurich, Switzerland, October 2009.
- [34] S. Lee and M. Younis, "Recovery from Multiple Simultaneous Failures in Wireless Sensor Networks using Minimum Steiner Tree", *Journal of Parallel and Distributed Systems*, Vol. 70, pp. 525-536, 2010.
- [35] S. Lee and M. Younis, "QoS-aware Relay Node Placement in a Segmented Wireless Sensor Network", in the *Proceedings of the IEEE International Conference on Communications (ICC 2009)*, Dresden, Germany, June 2009.
- [36] W Zhang, G. Xue and S. Misra, "Fault-Tolerant Relay Node Placement in Wireless Sensor Networks: Problems and Algorithms", in the *Proceeding of the 26th Annual Joint Conference of the IEEE Computer and Communications (INFOCOM'07)*, Anchorage, Alaska, May 2007.
- [37] X. Liu, L. Xiao, A. Kreling, and Y. Liu, "Optimizing Overlay Topology by Reducing Cut Vertices", in the *Proceedings of ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'06)*, Newport, Rhode Island, May 2006.
- [38] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks", *IEEE Transection on Wireless Communications*, Vol. 1, No. 4, October 2002, pp. 660-670.
- [39] A. Boukerche, X. Cheng, J. Linus, "A Performance Evaluation of a Novel Energy Aware Data-Centric Routing Algorithm in Wireless Sensor Networks", *Wireless Networks*, Vol. 11, No. 5, pp. 619-635, September 2005.
- [40] A. Savvides, C. C. Han, and M. Srivastava, "Dynamic Fine-Grained Localization in Ad-hoc Networks of Sensors", in the *Proceedings of the Annual ACM International*

Conference on Mobile Computing and Networking (MOBICOM'01), Rome, Italy, July 2001.

- [41] T. H. Cormen, C. E. Leiserson, R. L. Rivest, "Introduction to Algorithms 2nd Edition", *MIT Press and McGraw-Hill*, pp. 558–565 & pp. 570–576, 1990.
- [42] G. Gupta, M. Younis, "A Simulation Environment for Wireless Sensor Networks", *MS Project, Department of Computer Science and Electrical Engineering*, University of Maryland Baltimore County, USA, Fall 2003.
- [43] R. K. Jain, "The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling", *Wiley- InterScience*, New York, NY, April 1991.

VITA

Ameer Ahmed Abbasi is Pakistani citizen and born on 30-Aug-1976 in Karachi, Pakistan. He received his B.A and M.A degrees in computer technology from Sheikh Zayed Research Centre, University of Karachi, Pakistan in 1999 and 2000, respectively. He worked one and half years for the Usman Business Solution Inc. Karachi, Pakistan as a software engineer. In 2001, he moved to the Arab Cultural Institute (ACI) of Management & Computer Science, Dammam, Saudi Arabia. He worked with ACI as a senior instructor and taught computer science courses. At ACI, he was also acting as director of IT & data processing center. In 2011, he joined SofDigital Systems (SDS) as a first CEO. He is one of the co-founders of SDS. SofDigital Systems is a multidimensional and multinational organization that provides consultation and services in the field of software engineering, web technologies and IT infrastructure. In addition, he is actively participating in research in the fields of fault tolerance and topology management for mobile, ad-hoc and wireless sensor networks. He is serving in multiple review committees and has published several technical papers in refereed conferences and journals. Since 2008, he also is a part time graduate student with computer engineering department, King Fahd University of Petroleum & Minerals (KFUPM). His contact details are provided below:

Email addresses: *ameer@sofdigital.com*; *ameerabbasi25@gmail.com*

Mobile: +966-50-7936747, Home Phone: +966-3-8415069

Present Address: P.O. Box 8100, Dammam-31482, Saudi Arabia

Permanent Address: House # R-246, Khurram Abad, Landhi 1, Karachi 30, Pakistan