Recent Techniques in Node Failure Recovery in Wireless Sensor-Actor Networks

Nithya. S
PG Student
SMVEC, Puducherry

Dr. L. M. Varalakshmi
Associate professor
SMVEC, Puducherry

Abstract

Recent technological advances have lead to the emergence of distributed wireless sensor and actor networks (WSANs) which are capable of observing the physical world, processing the data, making decisions based on the observations and performing appropriate actions. These networks can be an integral part of systems such as battlefield surveillance and microclimate control in buildings, nuclear, biological and chemical attack detection, home automation and environmental monitoring. In most applications of Wireless sensor and actor network it is important to sustain connectivity among all actors at all times. When an actor fails the inter actor topology may get partitions into disjoint blocks and the application may be negatively impacted. Tolerating the actor failure and restoring the lost connectivity need to be performed while imposing the least overhead on the individual actors. In this paper a Least-Disruptive topology Repair (LeDiR) algorithm is proposed. LeDiR is to restore connectivity without extending the length of the shortest path among nodes compared to the prefailure topology. LeDiR is a localized and distributed algorithm that leverages existing route discovery activities in the network and imposes no additional prefailure communication overhead. The performance of LeDiR is analyzed and simulated in Network Simulator(NS2) Environment.

1. Introduction

In recent years, wireless sensor and actor networks (WSANs) have started to receive growing attention due to their potential in many real-life applications [1]. Such networks include miniaturized low-cost sensing nodes that are responsible for probing their surroundings and reporting their measurements to some actor nodes over wireless communication links. Actors process the sensed data, make decisions, and then perform the appropriate actions. The actor's response mainly depends on its capabilities and the application. For instance, actor can be used in lifting debris to search for survivors, extinguishing fires, chasing an intruder, etc.

Example of WSAN applications include facilitating/conducting urban search and rescue (USAR), detecting and countering pollution in coastal areas, performing in-situ oceanic studies of bird/fish migration and weather phenomena, detection and deterring of terrorist threats to ships in ports, destruction of mines in land and under water, and monitoring the environment for unusually high-level of radiation. 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 instance, in Urban Search and Rescue (USAR) applications in case of events such as fires, earthquakes, disasters, etc., the survivors can be in desperate need of oxygen gas, water or even some sort of medicine within a short period. Therefore, the actors should collaboratively decide the best possible solution in terms of the number of actors to employ, their traveling time, and distance to the survivor. This process requires that all the actors should be able to communicate in order to be aware of the current states of each other. To enable such communications, actors should form and maintain a connected inter-actor network at all times.

In such a scenario, an actor failure may cause the loss of multiple inter-actor communication links, partition the network if alternate paths among the affected actors are not available, and stop the actuation capabilities of the actor. Such a scenario will not only hinder the actors' collaboration but also may possibly risk the life of some survivors and thus have very negative consequences on the USAR application. Therefore, WSANs should be able to tolerate the failure of an actor and recover from it in a distributed, timely and energy efficient manner. 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 [2]. 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 nodes in separate partitions will not be able to reach each other to coordinate the recovery process. 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 relies on maintaining one- or two-hop neighbor lists and predetermines some criteria for the node's involvement in the recovery [3]–[5]. However, one-hop-based schemes often impose high node repositioning overhead, and the repaired inter-actor topology using two-hop schemes may differ significantly from its prefailure status.

For example, interaction among actors during a combat operation would require timeliness to accurately track and attack a fast moving target. A novel Least Disruptive topology Repair (LeDiR) algorithm is proposed. 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 of affected nodes is extended relative to its prefailure status. LeDiR is a localized and distributed algorithm that leverages existing route discovery activities in the network and imposes no additional prefailure communication overhead. When a node fails, its neighbors will individually consult their possibly incomplete routing table to decide on the appropriate course of actions and define their role in the recovery if any. If the failed node is critical to the network connectivity, i.e., a node whose failure causes the network to partition into disjoint blocks, the neighbor that belongs to the smallest block reacts. The performance of LeDiR is validated through simulation.

Section 2 describes the assumed system model and defines the considered problem. Section 3 gives an overview of related work. Section 4 explains LeDiR in detail. Section 5 describes the validation experiments and analyzes the simulation results. The paper is concluded in Section6.

2. System Model And Problem Statement

The WSAN network is composed of actors and sensors that are randomly deployed in an area. Actors are movable and have the capability to respond based on data collected by the sensors. All actors are assumed to have the same communication range. Since actors are more powerful than sensors, they typically have a longer communication range. After network deployment, a self-initialized phase is carried out by the whole nodes in the network. In this phase, each actor broadcasts a hello message with its identity and location. To cope with dynamic changes in the network, a heartbeat message is sent periodically by all actors. If an actor does not hear from its neighbour, a failure of that actor is assumed and the active actor has to take an immediate action.

The inter-actor topology can be modeled as a graph $G(N,E)$, where N is the number of actors and E is the number of edges. The actor's position plays a key role in the stability of the network connectivity. Actors can be classified into two types: cut-vertex and non cut-vertex. The failure of a cut-vertex actor partitions the network into isolated islands, while when a non cut-vertex actor fails; strong network connectivity is still maintained. For example, in Figure 1 *node2 and node8* are non cut-vertices while *node3 ,node5, node6 and node7* are cut-vertices. Therefore, to maintain the connectivity of the network, cut-vertex determination is important to react for node failures. Determining whether a node is a cut-vertex or not can be easily done by using depth first search trees (DFS). However, this approach requires flooding the whole network and can be costly in terms of the message overhead. Thus, LeDiR uses a distributed approach for such a purpose. Our LeDiR approach employs the concept of connected dominating set (CDS).

We assume that the actors can move on demand 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. The focus of this paper is on restoring strong connectivity at the level of interactor topology. It is assumed that a sensor node can reach atleast one actor over multihop paths and will not be affected if the actors have to change their positions. Thus, sensor nodes are not part of the recovery process. In the balance of this paper, actor and node are used interchangeably.

The impact of the actor's failure on the network topology can be very limited, e.g., a leaf node, or significant if the failed actor is a cut vertex. A node (vertex) in a graph is a cut vertex if its removal, along with all its edges, produces a graph with. more connected components (blocks) than the original graph.

For example, in Fig. 1, the network stays strongly connected after the loss of a leaf actor such as *node2*. Meanwhile, the failure of the cut vertex *node3* leaves nodes *5, 6, 7 and 9* isolated from the rest of the

network. In the rest of this paper, the terms cut vertex and critical node will be used interchangeably. To tolerate the failure of a cut vertex node, two methodologies can be identified: 1) precautionary and 2) realtime restoration. The precautionary methodology strives to provision fault tolerance by establishing a biconnected topology, where every pair of nodes *Ai* and *Aj* has two distinct paths with no common nodes other than *Ai* and *Aj* ; therefore, the network stays connected after a single node failure. However, provisioning such a level of connectivity may require the deployment of a large number of actors and can thus be impractical due to the high cost. In addition, it may constrain the mobility of actors and negatively affect application-level functionality.

.

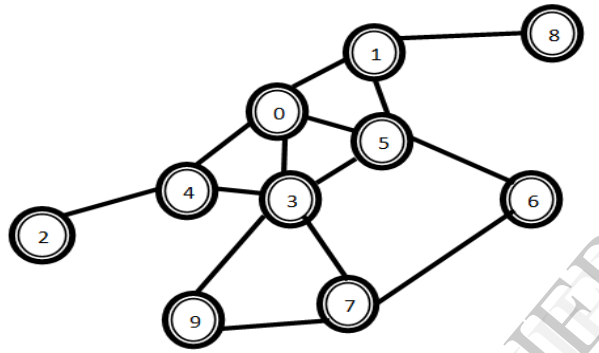

Fig. 1. Example one-connected inter-actor network. Nodes *3, 5, 6 and 7* are cut vertices whose failure leaves the network partitioned into two or multiple disjoint blocks.

On the other hand, real-time restoration implies a response only when a failure is detected. We argue that real-time restoration better suits WSANs since they are asynchronous and reactive in nature, where it is difficult to predict the location and scope of the failure. We further direct our attention to setups in which the interactions among actors are delay sensitive and the shortest data path between a pair of nodes should not get extended compared to its prefailure length.This paper assumes that only non-simultaneous node failures will take place in the network. To the best of our knowledge, most recovery schemes found in the literature assume no simultaneous faults. The focus of LeDiR is on nodes that are critical to 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 [2], [6]. 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 gets partitioned into disjoint blocks. To simplify the analysis, all nodes are assumed to have the same communication range.

However, our proposed algorithms do not require such assumption. In addition, the presentation of our work focuses on the algorithmic part of the recovery without focusing on the link layer issue. In general, any distributed medium access arbitration scheme would suffice. It is also assumed that a node would transmit at its maximum power to repair broken data routes before declaring a major connectivity problem and invoking LeDiR.

3. Related Work

While a number of schemes have been published recently for restoring network connectivity in partitioned WSANs, all of these schemes have focused on reestablishing severed links without considering the effect on the length of prefailure data paths. Some schemes recover the network by repositioning the existing nodes, whereas others carefully place additional relay nodes. The main idea is to identify and relocate some of the nodes. 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. DARA [3] and PADRA [7] also exploit cascaded motion in order to restore the connectivity. One of the neighbors of the failed node is picked to initiate the recovery process such that the movement overhead and the number of messages are minimized. While DARA designates the node with the least node degree as the recovery initiator, PADRA identifies a connected dominating set to determine a dominatee node. The dominatee does not directly move to the location of the failed node, rather a cascaded motion is pursued to share the burden. None of these approaches cares for the path length between nodes.

While LeDiR 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 in [3], [5], and [7] 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 [4] 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 lost link during the recovery will be re-established through cascaded relocation. The collective effect seems like the network topology is shrinking inward. The advantage of RIM is obviously the reduced overhead which is nonetheless provided at the expense of overreacting to failure of nodes that are not cut-vertices.

LeDiR utilizes the partial knowledge of a node about the network topology, gained during route discovery, to decide on which one participates and which one does not. No recovery-related explicit state update is required. On the other hand, some work on sensor relocation focuses on metrics other than connectivity, e.g., coverage [8] -[9], network longevity [10], and asset safety [11], or to self-spread the nodes after non-uniform deployment [6], [12], [13], which is not our focus in this paper.

4. Least-Disruptive Topology Repair

Upon the detection of network partitioning, LeDiR opts to identify the smallest block and limits the scope of the recovery to that block. The rationale is that fewer nodes will be involved and the overhead is minimized. As mentioned earlier, the goal for LeDiR is to restore connectivity without extending the length of the shortest path among nodes compared to the prefailure topology. LeDiR is a localized and distributed algorithm that leverages existing route discovery activities in the network and imposes no additional prefailure communication overhead. Before explaining how LeDiR works, it is important to point out the effect of contemporary recovery schemes on the path length between nodes.

The main idea for LeDiR is to pursue block movement instead of individual nodes in cascade. To limit the recovery overhead, in terms of the distance that the nodes collectivity travel, LeDiR identifies the smallest among the disjoint blocks. In addition, LeDiR opts to avoid the effect of the relocation on coverage and also limits the travel distance by stretching the links and moving a node only when it becomes unreachable to their neighbor. It is important to stress the fact that the focus of LeDiR is on nodes that are critical to network connectivity, e.g., cut vertices. The following major steps are used for LeDiR implementation.

4.1. Failure detection

Actors will periodically send heartbeat messages to their neighbors to ensure that they are functional, and also report changes to the one-hop neighbors.

Table I. Path predecessor matrix

	0	1	2	3	4	5	6	7	8	9
0	-	0	1	0	0	0	5	3	4	7
1	0	-	1	5	0	1	5	3	4	3
2	1	1	-	5	0	1	5	3	4	3
3	0	5	1	-	3	3	5	3	4	3
4	0	0	1	4	-	3	5	3	4	3
5	0	5	1	5	3	-	5	3	4	3
6	5	5	1	5	3	5	-	6	4	3
7	3	5	1	7	3	3	7	-	4	7
8	4	0	1	4	8	3	5	7	-	3
9	3	0	1	9	3	3	7	9	4	-

Missing heartbeat messages can be used to detect the failure of actors. Once a failure is detected in the neighborhood, the one-hop neighbors of the failed actor would determine the impact, i.e., whether the failed node is critical to network connectivity. This can be done using the SRT by executing the well-known depth-first search algorithm. Basically, a cut vertex F has to be on the shortest path between at least two neighbors of F. Consider Table I, which lists the entries of the SRT for the network topolgy in Fig. 1. After the failure of actor *node1*, which is a cut vertex, *node5* will check what nodes are reachable through *node1*, which are *node0* and *node8* in this example.

Checking the entries for *nodes0* and *node8* reveals that *node4*, *node3*, *node6* and *node9* will become consequently unreachable. The same is repeated and finally leads *node5* to conclude that only *node0* is reachable and *node1* 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-shaped topology, the node at the center is not a cut vertex, yet it serves on the shortest path 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.

4.2. Smallest block identification

LeDiR limits the relocation to nodes in the smallest disjoint block 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 direct 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.

4.3. Replacing faulty node

If node J is the neighbor of the failed node that belongs to the smallest block, J is considered the BC 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 a "child" if it is two hops away from the failed node, "grandchild" if three hops away from the failed node, and so on. The reason for selecting J to replace the faulty node is that the smallest block has the fewest nodes in case all nodes in the block have to move during the recovery. As will be shown later, the overhead and convergence time of LeDiR are 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 fits the characteristics of a BC, the closest actor to the faulty node would be picked as a BC. Any further ties will be resolved by selecting the actor with the least node degree. Finally, the node ID would be used to resolve the tie.

4.4. 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. LeDiR opts to avoid that by sustaining the existing links. Thus, if a child receives a message that the parent P is moving, the child then notifies its neighbors (grandchildren of node P) and travels directly toward the new location of P until it reconnects with its 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 [4]. Briefly, suppose a child C has two parents A and B that move toward the previous location of node J. As previously mentioned, node J already moved to replace the faulty node F, and as a result, nodes A and B get disconnected from node J. Now, nodes A and B would move toward the previous location of J until they are r/2 units away. Before moving, these parents inform the child C about their new locations. Node C uses the new locations of A and B to determine the slot to which it should relocate. Basically, node C will move to the closest point that lies within the communication ranges of A and B, which is the closest intersection point of the two circles of radius r and centered at A and B, respectively. It is worth to mention that since parents A and B move toward a single point, that is, the position of node J, they get closer to one another. Thus, if both can reach C before they move, i.e., C lies within their range, their communication range must overlap after the move since they get closer to one another. This observation also applies for more than two parent nodes since there must be an intersection point of two circles which lies within the communication ranges of all the moved nodes.

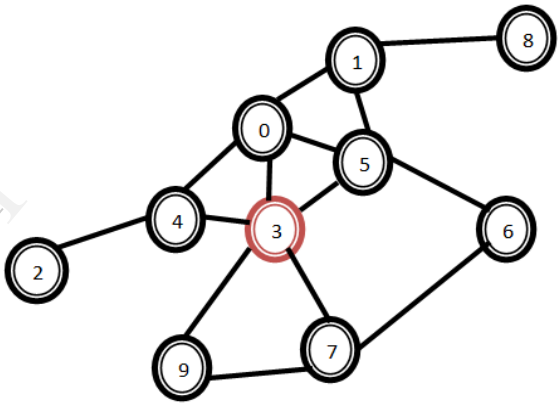

Fig. 2. Faulty node 3 is detected by missing of heartbeat messsage .

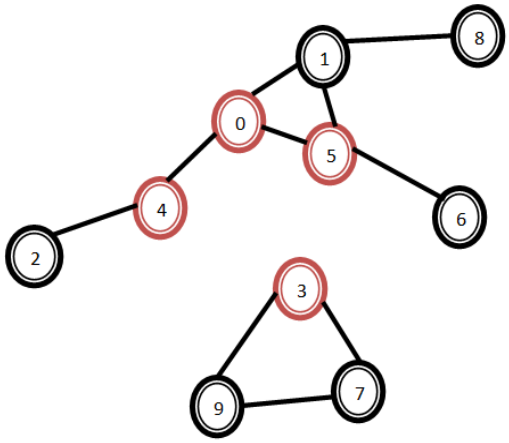

Fig. 3. Failure of actor node 3 which leads to disjoint the blocks and cause the network to partitioning.

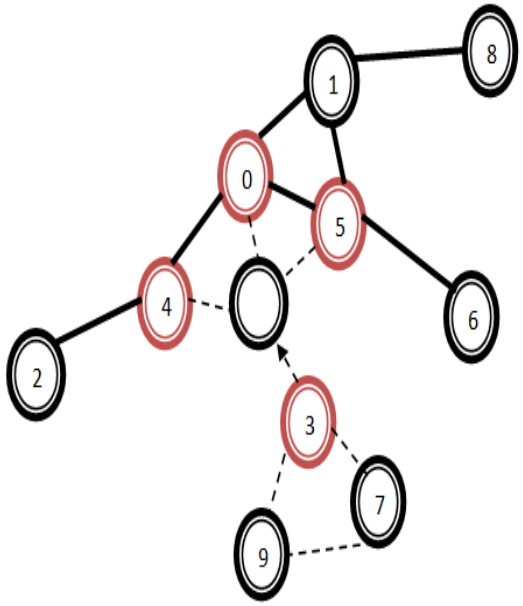

Fig. 4. How LeDiR restores connectivity after the failure of node A10 in the connected inter-actor topology.

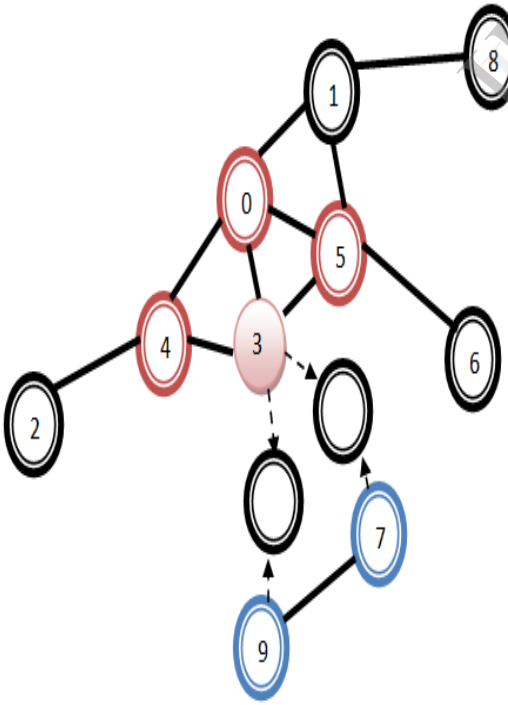

Fig. 5. Restoration of connectivity using LeDiR after the failure of node A10 in the connected inter-actor topology.

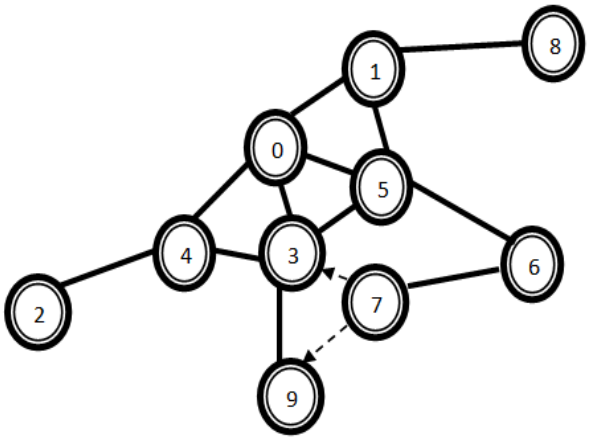

Fig. 6. Restoring the connectivity by replacing the position of node 3 by node 7.

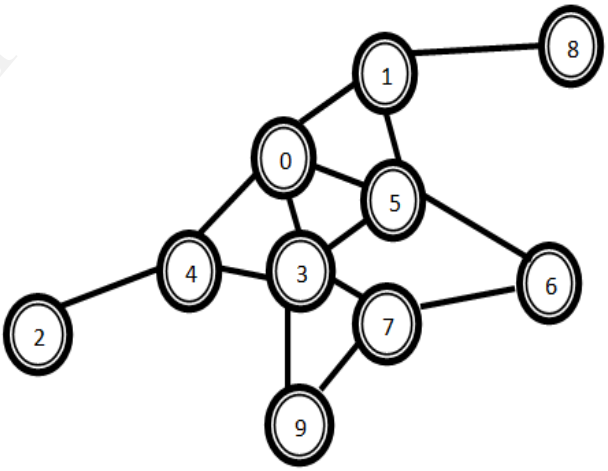

Fig. 7. LeDiR recovers the failure node with minimal topology changes in the network.

Fig. 2,3,4,5,6,7 shows an example for how LeDiR restores connectivity after the failure of node3. Obviously, *node3* is a cut vertex, and *node7* becomes the one-hop neighbor that belongs to the smallest block [see Fig. 4 and 5]. In Fig. 4, *node7* notifies its neighbors and moves to the position of *node3* to restore connectivity. Disconnected children, i.e., *node6* and *node9*, follow through to maintain communication link with *node7* [see Fig. 6]. Note that the objective of the children movement is to avoid any changes to the current routing table.

4.5. Distributed LeDiR Implementation

The foregoing discussion 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 the 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 that a route has yet to be discovered. In general, a partially populated SRT can raise the following three issues for a distributed implementation of LeDiR: 1) A potential BC actor does not realize that its failed neighbor is a critical node; 2) every neighbor of the faulty node assumes that it is not part of the smallest block leaving the network topology unrepaired; and 3) more than one neighbor in different blocks step forward as BC. In the balance of this section, we discuss how LeDiR addresses these issues.

Let α be the percentage of entries, i.e., routes between actor pair (i, j), that each node has acquired over time. Hereafter, we will call this α as confidence level (CL). For example, if 50% entries of the node's Ai routing table are filled, we say node Ai has 50% CL. Since every node may potentially have different CL from others, upon the detection of a node failure,applying depth-first search at the neighboring nodes may yield an inconsistent assessment of the impact of the node loss on the network connectivity and on which actor is the BC for leading the recovery. In addition, the operation in WSAN is collaborative in nature, and an actor usually communicates with many others; thus, the routing table would not be sparse or at least will include the important routes; in particular, the neighbors of a cut vertex would have more populated SRT compared 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 two-hop information to boost the fidelity of the assessment [5], [14] and mitigate the effect of the missing entries in the SRT. 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 [5]. 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 BC and performs LeDiR successfully, i.e., proceeds to replace the faulty node. Children would follow BC to maintain connectivity, and so on.

The foregoing second and third issues are related to determining the BC, 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 node7 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 node3. Since the neighbors of node3 cannot reach each other, a partially populated SRT may lead to a deadlock, with none of the neighbors of node3 responding to the failure and leaving the network disconnected. To handle this issue, LeDiR imposes a timeout after which the neighbor(s) belonging to the second largest block will move. This time, multiple neighbors may be potentially moving toward node3. To avoid having more than one actor replacing node3, LeDiR requires these nodes to broadcast messages with their ID so that they pause as soon as reaching other neighbors of node3 that happen to be in a different block. The pause time would allow these neighbors to negotiate and pick the BC to continue on to the position of node3.

5. Performance Analysis

LeDiR is validated through the simulation. This section discusses the simulation environment and experimental results.

5.1. Simulation Environment and Performance Metrics

The experiments are performed on a NS2(Network Simulator 2). In the experiments, we have created

connected topologies consisting of varying number of actors (1 to 10) with fixed transmission range (r = 100m). In addition, we run simulations with fixed nodes count (10 actors) while varying communication range (200m to 450m). After identifying the cut-vertices in the generated topology, one of them is designated at random to be the faulty node. Floyd-Warshall algorithm is used to form the SRT. This implicitly implies that every node is aware of the entire network topology.

The following parameters are used to vary the characteristics of the WSAN topology in the different experiments:

- *Communication range (r)*: All actors have the same communication range *r*. The value of *r* affects the initial WSAN topology. While a small *r* creates a sparse topology, a large *r* boosts the overall network connectivity.
- *Number of Deployed Actors (N)*: This parameter affects the node density and the WSAN connectivity. Increasing the value of *N* would affect the node density and thus WSAN topology would become highly-connected.

5.2. Simulation Results

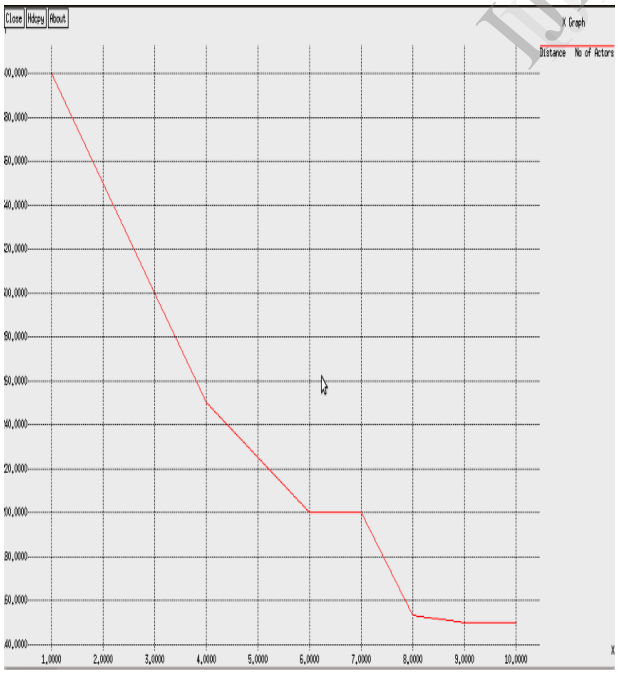

Fig. 8. Minimum number of actors used for recovering the faulty node by effective implementation of LeDiR algorithm.

6. Conclusion

This paper has tackled an important problem in mission critical WSANs; that is sustaining network connectivity 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. The performance of LeDiR, in terms of the travelled distance and minimum number of actors has been validated through simulation. The results have demonstrated that LeDiR is almost insensitive to the variation in the commutations range. LeDiR also works very well in dense networks and yields closed to optimal performance even when nodes are partially aware of the network topology.

7. References

[1] I. F. Akyildiz and I. H. Kasimoglu, "Wireless Sensor and actornetworks: Research Challenges," Elsevier Ad hoc Network Journal, Vol. 2, pp. 351-367, 2004.
[2]. M. Younis and K. Akkaya, "Strategies and techniques for node placement
in wireless sensor networks: A survey," J. Ad Hoc Netw., vol. 6, no. 4,pp. 621–655, Jun. 2008.
[3] A. Abbasi, M. Younis, and K. Akkaya, "Movement-assisted connectivity
restoration in wireless sensor and actor networks," IEEE Trans. Parallel Distrib. Syst., vol. 20, no. 9, pp. 1366–1379, Sep. 2009.
[4]. M. Younis, S. Lee, S. Gupta, and K. Fisher, "A localized self-healing algorithm for networks of moveable sensor nodes," in *Proc.* IEEE GLOBECOM, New Orleans, LA, Nov. 2008, pp. 1–5.
[5]. K. Akkaya, F. Senel, A. Thimmapuram, and S. Uludag, "Distributed recovery from network partitioning in movable sensor/actor networks via controlled mobility," IEEE Trans. Comput., vol. 59, no. 2, pp. 258–271, Feb. 2010.
[6]. K. Akkaya and M. Younis, "COLA: A coverage and latency aware actor placement for wireless sensor and actor networks," in Proc. IEEE VTC, Montreal, QC, Canada, Sep. 2006, pp. 1–5.
[7]. A. Youssef, A. Agrawala, and M. Younis, "Accurate anchor-free localization in wireless sensor networks," in Proc. 1st IEEE Workshop Inf. Assurance Wireless Sensor Netw., Phoenix, AZ, Apr. 2005.
[8]. S. Yang, M. Li, and J.Wu, "Scan-based movement-assisted sensor deployment methods in wireless sensor networks," IEEE Trans. Parallel Distrib. Syst., vol. 18, no. 8, pp. 1108–1121, Aug. 2007.
[9]. R.-S. Chang and S.-H. Wang, "Self-deployment by density control in sensor networks," IEEE Trans. Veh. Technol., vol. 57, no. 3, pp. 1745– 1755, May 2008.

[10]. K. Dasgupta, M. Kukreja, and K. Kalpakis, "Topology-aware placement and role assignment for energy-efficient information gathering in sensor networks," in Proc. 8th ISCC, Kemer-Antalya, Turkey, Jun. 2003, pp. 341–348.
[11]. W. Youssef, M. Younis, and K. Akkaya, "An intelligent safety-aware gateway relocation scheme for wireless sensor networks," in Proc. ICC, Istanbul, Turkey, Jun. 2006, pp. 3396–3401.
[12]. H. Liu, X. Chu, Y.-W. Leung, and R. Du, "Simple movement control algorithm for bi-connectivity in robotic sensor networks," IEEE J. Sel. Areas Commun., vol. 28, no. 7, pp. 994–1005, Sep. 2010.
[13]. G. Tan, S. A. Jarvis, and A.-M. Kermarrec, "Connectivity-guaranteed and obstacle-adaptive deployment schemes for mobile sensor networks," in Proc. 28th ICDCS, Beijing, China, Jun. 2008, pp. 429–437.
[14]. X. Liu, L. Xiao, A. Kreling, and Y. Liu, "Optimizing overlay topology by reducing cut vertices," in Proc. ACM Workshop Netw. Operating Syst. Support Digital Audio Video, Newport, RI, May 2006, pp. 1–6.