Review on Improved Fault Tolerance in WSAN with Minimal Topology Changes

Perraju P. Tetali
PG Department of Computer Science and Engineering
WCEM, RTM Nagpur University
Nagpur, India.
tetali_raju@yahoo.com

Garima Singh
PG Department of Computer Science and EngineeringWCEM, RTM Nagpur University
Nagpur, India.
garima11makhija21@gmail.com

***Abstract*—In wireless sensor-actor networks, sensors review their surroundings and forward their data to actor nodes. Actors collectively respond to achieve predefined application mission. Since actors have to coordinate their operation, it is necessary to maintain a strongly connected network topology at all times. Moreover, the length of the inter-actor communication paths may be constrained to meet latency requirements. However, a failure of an actor may cause the network to partition into disjoint blocks and would, thus, violate such a connectivity goal. One of the effective recovery methodologies is to autonomously reposition a subset of the actor nodes to restore connectivity. Contemporary recovery schemes either impose high node relocation overhead or extend some of the inter-actor data paths. This paper overcomes these shortcomings and presents a Least-Disruptive Topology Repair (LDTR) algorithm. LDTR relies on the local view of a node about the network to devise a recovery plan that relocates the least number of nodes and ensures that no path between any pair of nodes is extended. LDTR 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 LDTR is analyzed mathematically and validated via extensive simulation experiments.**

***Index Terms*—Fault tolerance, network recovery, topology man-agement, wireless sensor-actor network (WSAN).**

I. 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 includeminiaturized 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.

This paper considers the connectivity restoration problem subject to path length constraints. Basically, in some applications, such as combat robotic networks and search-and-rescue operation, timely coordination among the actors is required, and extending the shortest path between two actors as a side effect of the recovery process would not be acceptable. For example, interaction among actors during a combat operation would require timeliness to accurately track and attack a fast moving target.

A Least Disruptive topology Repair (LDTR) algorithm is proposed to solve the problem.LDTR 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. LDTR 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 LDTR is validated both analytically and through simulation. The simulation results demonstrate that LDTR outperforms existing schemes in terms of communication and relocation overhead.

The next section describes the assumed system model and defines the considered problem. Section III gives an overview of related work. Section IV explains LDTR in detail. The paper is concluded in Section V.

II. SYSTEM MODEL AND PROBLEM STATEMENT

As mentioned earlier, a WSAN involves two types of nodes: 1) sensors and 2) actors. Sensors are inexpensive and highly constrained in energy and processing capacity. On the other hand, actors are more capable nodes with relatively more onboard energy supply and richer computation and communication resources. However, the transmission range of actors is finite and significantly less than the dimensions of the deployment area. Although actors can theoretically reach each other via a satellite channel, the frequent inter-actor interaction required by WSAN applications would make the often intermittent satellite links unsuitable. It is thus necessary for actors to rely mostly on contemporary terrestrial radio links for coordination among themselves. Upon deployment, actors are assumed to discover each other and form a one-connected network using some of the existing techniques. An actor employs ranging technologies and localization techniques to determine its position relative to its neighbor. We assume that the actors can move on

demand to perform tasks on larger areas or to enhance the interactor 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 strongconnectivity at the level of inter-actor topology. It is assumed that a sensor node can reach at least 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 A21 or a nonleaf node like A5. Meanwhile, the failure of the cut vertex A0 leaves nodes A4, A5, and A6 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, twomethodologiescan be

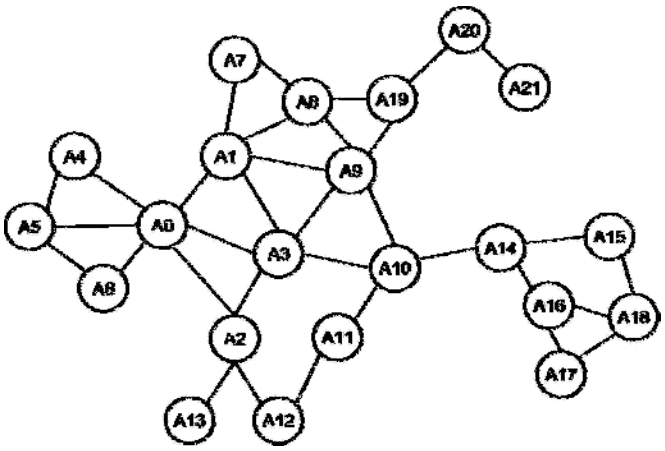

Fig. 1.Example one-connected inter-actor network. Nodes A0, A10, A14, and A19 are cut vertices whose failure leaves the network partitioned into two or multiple disjoint blocks.

identified: 1) precautionary and 2) real-time 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. 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 nonsimultaneous 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 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 two simultaneous faults is p2, p3 for three, etc. With p being a small fraction, the probability of multiple faults diminishes. In addition, the focus of LDTR 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. 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 LDTR.

III. RELATED WORK

A number of schemes have recently been proposed for restoring network connectivity in partitioned WSANs [2]. 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. On the other hand, some work on sensor relocation focuses on metrics other than connectivity, e.g., coverage, network longevity, and asset safety, or to self-spread the nodes after nonuniform deployment [6], which is not our focus in this paper.

A. Recovery Through Node Repositioning

The main idea of this category of recovery schemes is to reposition some of the healthy nodes in the network to rein-state strong connectivity. LDTR fits in this category.

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 Distributed Actor Recovery Algorithm (DARA) [3] and PArtition Detection and Recovery Algorithm (PADRA) require every node to maintain a list of their two-hop neighbors and determine the scope of the recovery by checking whether the failed node is a cut vertex. DARA pursues a probabilistic scheme to identify cut vertices. A best candidate (BC) is selected from the one-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 [5], the focus is also 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 relink these blocks, the closest nodes are moved toward 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 LDTR also employs cascaded relocation, the criteria for selecting the lead node and other participants are different.

To ensure that the recovery process converges in an efficient way, the approaches in [3], [5] require each node in the network to be aware of its two-hop neighbors. The availability of two-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. Recovery through Inward Motion (RIM) [4] and Least Distance Movement Recovery (LDMR), on the other hand, defy that assumption and base the recovery process on the knowledge of direct, i.e., one-hop, neighbors. Simply, the neighbors of a node F detect that F has failed, and then move toward F until they can reach each other directly. In RIM, any lost link during the recovery will be reestablished through cascaded relocation. The collective effect seems like the network topology is shrinking inward. LDMR avoids the cascaded relocation by sending messages to find the replacement for the neighbors of F after they move. The advantage of RIM and LDMR is obviously the reduced prefailure communication overhead that is nonetheless provided at the expense of overreacting to failure of uncritical nodes. LDTR utilizes the partial knowledge of a node about the network topology, gained during route discovery, to decide on which node participates and which node does not. No recoveryrelated explicit state update is required.

Unlike LDTR, Connectivity Restoration through node Rearrangement (CRR) 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 onehop neighbors of the faulty node are relocated at these points. In case the number of onehop neighbors is not enough, the approach progresses as the DARA approach, as previously discussed. These approaches would fit more of a planned rather than a reactive recovery scenario, as targeted by LDTR.

Upon the detection of network partitioning, LDTR 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 reduced. Obviously, the goal of the block movement is to restore network biconnectivity rather than repairing a disjointed network. Some prior work cared about the coverage hole in the network when a node fails rather than connectivity.

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 connectivitylimited focus of the recovery. Unlike the approaches previously discussed, Coverage Conscious Connectivity Restoration (C3R) tackles the loss of both coverage and connectivity. C3R involves onehop neighbors of the faulty node in the recovery process. All the one-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 originally covered spots. Finally, node relocation has been pursued to optimize network performance, including boosting connectivity, not necessarily to deal with node failure. A survey of such work can be found in [2].

B. Recovery by Placement of Relay Nodes

The foregoing 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 and Distributed algorithm for Optimized Relay node placement using Minimum Steiner tree (DORMS) opt

to not only reestablish the network connectivity but also achieve a certain quality in the 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 toward 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 the area. Then, adjacent branches are further optimized by forming a Steiner tree for connecting two segments and the center node to reduce the required relay count.

Meanwhile, intersegment connectivity ought to maintain some level of quality of service (QoS) while placing the least number of relay nodes. The proposed approach initially models the deployed area as a grid with equalsized 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 cellbased least cost paths and populating nodes along these paths. On the other hand, Zhang et al. form a biconnected intersegment topology by placing redundant nodes so that the failure of a node can be tolerated and the network operation continues without interruption. Al-Turjman et al. model the connectivity restoration as a node placement problem on a grid and reposition the deployed nodes to meet varying requirements on the intersegment traffic. As mentioned earlier, LDTR is a reactive scheme that opts to restore connectivity while imposing the least travel overhead and in a distributed manner.

IV. LEASTDISRUPTIVE TOPOLOGY REPAIR

As mentioned earlier, the goal for LDTR is to restore connectivity without extending the length of the shortest path among nodes compared to the prefailure topology. In this section, we first give an overview of LDTR as a centralized solution and then explain the distributed implementation.

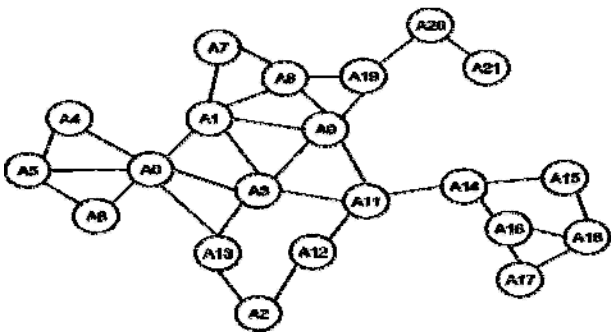

Fig. 2. How DARA [3] restores connectivity after the failure of node A10 in the connected inter-actor topology of Fig. 1.

*A. Problem and Solution Analysis*Before explaining how LDTR works, it is important to point out the effect of contemporary recovery schemes on the path length between nodes. Let us consider Fig. 1 and assume that node A10 fails. Connectivity restoration schemes that exploit node repositioning will replace A10 with one of its neighbors. For example, DARA picks the neighbor with the least degree to limit the scope of relocation. Thus, A11 relocates to the position of A10. The connectivity restoration process will be repeated with repositioning A12 to replace A11, followed by relocating A2 to where A12 was. Finally, A13 replaces A2. The resulting topology is shown in Fig. 2. While A0 and A3 were directly reachable to A2 before the failure, the repaired topology in Fig. 2 makes the shortest path one hop longer by involving A13. As mentioned in Section I, this will not be acceptable for delay sensitive applications. LDTR opts to avoid such a scenario by sustaining or even shortening the prefailure path lengths.

The main idea for LDTR 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, LDTR identifies the smallest among the disjoint blocks. For the previous example when A10 fails, LDTR will only involve the block of node A14. In addition, LDTR 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. As mentioned in Section II, it is assumed that no simultaneous node failures would take place. It is important to stress the fact that the focus of LDTR is on nodes that are critical to network connectivity, e.g., cut vertices.

The following highlights the major steps.

1. Failure detection: Actors will periodically send heartbeat messages to their neighbors to ensure that they are functional, and also report changes to the onehop neighbors. 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. After the failure of actor A19, which is a cut vertex, node A20 will check what nodes are reachable through A19, which are A8 and A9 in this example. Checking the entries for nodes A8 and A9 reveals that A1, A3, A7, and A10 will become consequently unreachable. The same is repeated and finally leads node A20 to conclude that only A21 is reachable and A19 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 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.

*2. Smallest block identification:*LDTR 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. For example, let us again consider the network topology provided in Fig. 1 and assume that node A19 failed. When nodes A8, A9, and A20, the one-hop neighbors of A19, confirm that A19 is indeed a cut vertex (critical node), they will be able to identify the disjoint blocks. For A20, the analysis of the cut vertex detection step discussed previously will conclude that A20 can reach only A21, and thus, A20 and A21 constitute a block. Now, A20 would check the column of A19 and find out that A8 and A9 are the other direct neighbors of A19. Node A20 will then repeat the analysis and identify the other disjoint block(s) and determine the smallest block after A19 fails. Now, A20 will lead the recovery effort if it happens to belong to the smallest block, which is the case in this example. Nodes A8 and A9 will perform the same analysis and conclude that they are not part of the smallest block.

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 LDTR 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.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. For example, in Fig. 2, the path between A2 and A3 get extended because A2 lost its link to A12 after A12 had moved. LDTR 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. It has been proven in [6] that this relocation scheme sustains existing links in the connected component (block).

V. CONCLUSION

In recent years, wireless sensor and actor (actuator) networks (WSANs) have started to receive growing attention due to their potential in many real-life applications. This paper has tackled an important problem in mission critical WSANs, that is, reestablishing network connectivity after node failure without extending the length of data paths. We have proposed a new distributed LDTRalgorithm that restores connectivity by careful repositioning of nodes. LDTR relies only on the local view of the network and does not impose prefailure overhead. The performance of LDTR has been validated through hard analysis and extensive simulation experiments. The experiments have also compared LDTR with a centralized version and to

contemporary solutions in the literature. The results have demonstrated that LDTR is almost insensitive to the variation in the communication range. LDTR also works very well in dense networks and yields close to optimal performance even when nodes are partially aware of the network topology.LDTR 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. Considering such a problem with collocated node failure is more complex and challenging in nature. In the future, we plan to investigate this issue. Our future plan also includes factoring in coverage and ongoing application tasks in the recovery process and developing a method for evaluating the various failure recovery schemes.

REFERENCES

[1]. M.Sureka ,S.L.JanyShabu."Pre-Recover from a Node Failure in Ad hoc Network Using Reactive Protocols", International Journal of Engineering Trends and Technology(IJETT), V8(5),262-266 February 2014. ISSN:2231-5381. www.ijettjournal.org. published by seventh sense research group.

[2]. Alfadhly, U. Baroudi, and M. Younis, “Least distance movement recovery approach for large scale wireless sensor-actor networks,” in Proc.WorkshopFedSenS, Istanbul, Turkey, Jul. 2011.

[3]. F. Senel, M. Younis, and K. Akkaya, “Bio-inspired relay node placement heuristics for repairing damaged wireless sensor networks,” IEEE Trans. Veh. Technol., vol. 60, no. 4, pp. 1835–1848, May 2011.

[4]. S. Lee and M. Younis, “Recovery from multiple simultaneous failures in wireless sensor networks using minimum Steiner tree,” J. Parallel Distrib. Comput., vol. 70, no. 5, pp. 525–536, May 2010.

[5]. K. Akkaya, A. Thimmapuram, F. Senel, and S. Uludag, “Distributed recovery of actor failures in wireless sensor and actor networks,” in Proc. IEEE WCNC, Las Vegas, NV, Mar. 2008, pp. 2480–2485.

[6]. 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.

[7]. 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.