An Effective Approach for Recovering From Simultaneous Node Failures in Wireless Sensor Networks

R.Hema,
Asst.Prof,Dept.of CSE
Annai Vailankanni College of Engineering.
hema25me@gmail.com

M.Kalai Amutha
Asst.Prof,Dept.of CSE
Annai Vailankanni College of Engineering.
mnkalai27@gmail.com

Abstract- In wireless sensor-actor networks, sensors probe their surroundings and forward their data to actor nodes. Actors collaboratively 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. Failure of one or multiple actors may partition the inter-actor network into disjoint segments, and thus hinders the network operation. Autonomous detection and rapid recovery procedures are highly desirable in such a case. 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 extended version of DCR named RAM, to handle one possible case of a multi-actor failure with Least-Disruptive topology Repair (LeDiR) algorithm for minimal topological changes. Upon failure detection, the backup actor initiates a recovery process that relocates the least number of nodes.

Index Terms— *Network recovery, Wireless Sensor-Actor Network (WSAN), Connectivity restoration.*

*****

I. INTRODUCTION

Recent years Wireless Sensor and Actor Networks are gaining growing interest because of their suitability for mission critical applications that require autonomous and intelligent interaction with the environment. Examples of these applications include forest fire monitoring, disaster management, search and rescue, security surveillance, battlefield reconnaissance, space exploration, coast and border protection, etc.

WSANs consist of numerous miniaturized stationary sensors and fewer mobile actors. The sensors serve as wireless data acquisition devices for the more powerful actor nodes that process the sensor readings and put forward an appropriate numerous miniaturized stationary sensors and fewer mobile actors.

The sensors serve as wireless data acquisition devices for the more powerful actor nodes that process the sensor readings and put forward an appropriate response. For example, sensors may detect a fire and trigger a response from an actor that has an extinguisher. Robots and unmanned vehicles are example actors in practice [1]. Actors work autonomously and collaboratively to achieve the application mission. Given the collaborative actors' operation, a strongly connected inter-actor network topology would be required at all times. Failure of one or multiple nodes may partition the inter-actor network into disjoint segments. Consequently, an inter-actor interaction may cease and the network becomes incapable of delivering a timely response to a serious event. Therefore, recovery from an actor failure is of utmost importance.

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]. 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 re-quire every node to maintain partial knowledge of the network state. To avoid the excessive state-update overhead and to ex-pedite 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].

Unlike prior work, this paper considers the connectivity restoration problem subject to path length constraints. In some applications, 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.

Most of the existing approaches in the literature are purely reactive with the recovery process initiated once the failure of ''F'' is detected. The main idea is replace the failed node ''F'' with one of its neighbors or move those neighbors inward to autonomously mend severed topology in the vicinity of F.

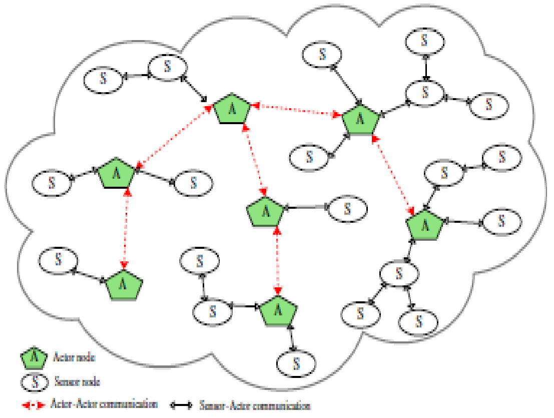


Fig:1 An Example wireless sensor and actor network setup

In this paper, we present a novel distributed partitioning Detection and Connectivity Restoration (DCR) algorithm, which proactively determines potential critical actors and assign backup nodes in order to rapidly repair the topology. In DCR, each actor proactively assesses its criticality, i.e., being a cut-vertex in the network topology, in a distributed manner based on the local information. The backup actor continuously monitors its primary for possible failure. Once the failure is detected, the backup initiates a recovery process by replacing the primary so that the connectivity is restored. The algorithm is recursively executed until all actors become strongly connected. DCR assumes single critical actor failure at time and no other node fails during the recovery process.

We extend DCR to address one scenario of the multi-node failure when no more than two of the failed actors are adjacent.RAM identifies critical actors and designates for them distinct backups. During the recovery process a novel Least-Disruptive topology Repair (LeDiR) algorithm is being executed to have minimal topological changes in the network.

The next section describes the assumed system model and defines the considered problem. Section III gives an overview of related work. The proposed DCR and RAM algorithms are detailed in Section IV. Section V explains LeDiR in detail Section VI presents the proposed recovery algorithms for RAM & LeDiR. The paper is concluded in Section VII.

II. SYSTEM MODEL AND PROBLEM STATEMENT

As mentioned earlier, a WSAN involves two types of nodes: 1) sensors and 2) actors. Sensors detect and report events of interest to one or multiple actors. Actors receive reports from sensors, process and collaborate with each other to plan an optimal coordinated response. It is thus necessary for actors to rely mostly on contemporary terrestrial radio links for coordination among themselves. Both sensors and actors are deployed randomly in an area of interest. After deployment, actors are assumed to discover each other and form a connected inter-actor network using some of the existing techniques such as [6]. An actor is assumed to be able move on demand and such relocation does not affect sensor actor connectivity. The action range of an actor refers to the maximum area in which an actor can cover and is assumed to be equal for all actors. Each actor maintains a list of direct (1-hop) neighbors and exchanges heartbeat messages with them to update its status.

DCR and RAM are suited for applications in which line-of-sight links are available between actors that fall in the communication range of each other. The impact of an actor's failure depends on the position of that actor in the network topology. A node is said to be critical, cut-vertex in graph theory terminology, if its removal partitions the network into disjoint segments. The failure of one or multiple critical actors not only affects the actor coverage but significantly impacts inter-actor connectivity. For example, consider a network topology depicted in Fig. 2. Losing a leaf/non-critical node, such as G does not affect inter-actor connectivity. Meanwhile, the failure of a critical node such as F partitions the network into disjoint blocks. This paper focuses on restoring inter-actor connectivity lost due to failure of one or multiple adjacent critical actors.

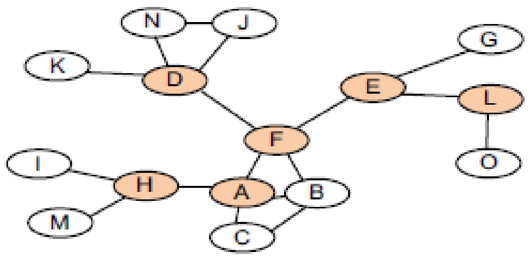


Fig 2: An Example of connected inter-actor network topology

In order to tolerate critical node failure, three methodologies can be identified: (i) proactive, (ii) reactive and (iii) hybrid. Proactive approaches establish and maintain bi-connected topology in order to provide fault tolerance. , in reactive approaches the network responds only when a failure occurs. Therefore, reactive approaches might not be suitable for time-critical applications. In hybrid approaches pre-failure planning is pursued in order to increase the efficiency of the recovery.

DCR uses a localized scheme to identify critical actors and designate backups for them. DCR considers one failure at a time and no other node fails during the recovery. We extend DCR to handle one possible scenario of multiple simultaneous failures of actors by RAM with minimal topological changes using LeDiR.

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.

Like our proposed DCR algorithm, DARA [7] strives to restore connectivity lost due to failure of cut-vertex. However, DARA requires more network state in order to ensure convergence. Meanwhile, in PADRA [8], identify a connected dominating set (CDS) of the whole network in order to detect cut-vertices. Although, they use a distributed algorithm, their solution still requires 2-hop neighbor's information that increases messaging overhead. Another work proposed in [9] also uses 2-hop information to detect cut-vertices. The proposed DCR algorithm relies only on 1-hop information and reduces the communication overhead.

Although RIM [10], C^{3}R [11] and VCR [12] use 1-hop neighbor information to restore connectivity, they are purely reactive and do not differentiate between critical and non-critical nodes. Whereas, DCR is a hybrid algorithm that proactively identifies critical nodes and designates for them appropriate backups. The existing work on simultaneous node failure recovery proposed in [8] is a mutual exclusion mechanism called [13] in order to handle multiple simultaneous failures in a localized manner.

Our proposed approach differs from MPADRA in multiple aspects. Whereas, our approach only requires 1-hop information and each critical node has only one backup to handle its failure.

IV. PARTITION DETECTION AND CONNECTIVITY RESTORATION

As mentioned earlier, hybrid algorithms better suits time-sensitive applications that require a rapid recovery. The proposed DCR algorithm is hybrid in the sense it consists of two parts, i.e. proactive and reactive. Once critical nodes (primary) are determined, they select and designate an appropriate neighbor (backup) to handle their failure when such contingency arises in the future. Each backup starts monitoring its primary through HEARTBEATS. The backup replaces the primary and cascaded relocations are performed until the recovery is complete. The detailed algorithm is described in the balance of this section.

4.1. Identifying critical actors

As described earlier, the failure of critical actor divides the inter-actor network into disjoint segments. DCR and RAM opt to identify a backup for each of these critical actors. Therefore, DCR and RAM employ a simple localized cut-vertex detection procedure that only requires 1-hop positional information to detect critical nodes. The procedure is based on [14] and runs on each node in a distributed manner to determine locally whether a node is critical or not. DCR and RAM employ a simple localized cut-vertex detection procedure that only requires 1-hop positional information to detect critical nodes. The procedure is based on [14] and runs on each node in a distributed manner to determine locally whether a node is critical or not.

Each actor determines locally whether it is critical or not based on neighbor's position information. It calculates the distance between neighbors based on their positions. If the distance is less than their communication range, the actor is considered non-critical because neighbors would stay connected without it. On the other hand, if the 1-hop neighbors of an actor can be partitioned into more than one segment, the actor is 1-hop critical.

For instance, Fig. 3 shows a localized scope of non-critical node A and critical node F. Nodes B, C, D, and E are 1-hop neighbors of node A as shown in Fig. 3(a). Node A is 1-hop positional non-critical because its neighbors remain connected without A. On the other hand, neighbors of F can be divided in to two sub graphs i.e. {B, C} and {G, H, I}. Therefore, F is 1-hop positional critical as illustrated in Fig. 3(b). Furthermore, leaf nodes such as I are detected as non-critical, since there failure does not inflict inter-actor connectivity. Again, 1-hop positional critical nodes are not indeed cut-vertices all the time; obviously, the opposite is true. However, DCR and RAM pursue such approximate state determination in order to cut on messaging overhead.

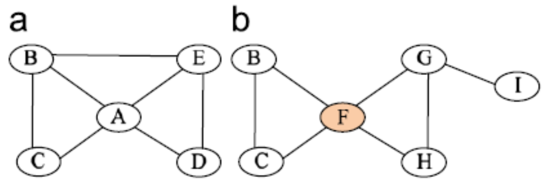


Fig 3:A Segment of an inter actor network showing 1-hop positional: (a) Critical and (b)non-critical actors

4.2. Recovery from single critical actor failure

This subsection details the DCR algorithm that is designed to recover from the failure of a critical actor. The details of the algorithm are in the following subsection.

4.2.1. Backup selection and primary monitoring

Once the critical actors (primary) are identified, the next step is to select and designate appropriate neighbors as backups. The purpose of the pre-nomination of backup nodes is to instantaneously react to the failure of critical actors and avoid the possible network partitioning caused by such a failure.

Selection of backup: The actors maintain minimum state information (i.e. 1-hop neighbors) to avoid extra overhead of messaging. Since, neighbors become disconnected when a critical actor fails, backup actors are determined and notified before a failure of critical nodes takes place. For DCR, a node can serve as backup for multiple actors; this will be constrained in RAM, as we discuss later. The selection of a backup among 1-hop neighbors is based on the following ordered criteria:

(a) *Travel feasibility*: An actor ''A'' that can relocate to the position of ''F'' due to the presence of physical constraints.

(b) *Neighbor actor status* (NAS): As discussed above, each actor determines whether it is critical or non-critical. A non-critical neighbor actor is preferred to serve as backup.

(c) *Actor degree (AD).* If a non-critical node is not available in the neighborhood, DCR prefers to choose a strongly connected critical node (with high degree) because there is more probability to have non-critical nodes in the neighborhood.

(d) *Inter-actor distance (ID):* A close backup actor is preferred in order to reduce the movement overhead and shorten the recovery time.

Once the critical actors (primary) are identified, they appoint appropriate backups to handle their failure. Like DCR, RAM also maintains minimum network state information, namely for 1-hop neighbors, in order to cut on the messaging overhead. However, RAM imposes additional constraints while choosing a backup in order to ensure convergence and avoid the creation of another network partitioning.

In addition to the criteria applied by DCR, the selection of a backup among 1-hop neighbors considers the Neighbor Criticality and Availability (NCA). Basically each actor maintains a state whether it is already engaged as a backup by some node and its position in the network topology, i.e., critical/ non-critical. This state information is shared with neighbors through periodic heartbeat messages. The following are applied when selecting a backup:

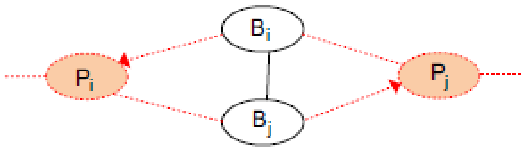


Fig 4: Illustrating the challenges in handling multiple simultaneous failures where moving two non-critical partitions the network.

- When a critical actor chooses a backup, it prefers a non-critical node that is not serving another primary. In other words, a non-critical node cannot have more than one primary as long as another free non-critical node is available in the neighbor-hood. A critical node is restrained from choosing a non-critical node as backup that is already designated for another actor. This is to ensure recovery in case two adjacent actors fail simultaneously.
- A critical node can be chosen as a backup only if it is not already appointed by some other node. Moreover, two adja-cent critical actors cannot serve each other as backup simultaneously. This will ensure that there will be some backup node to recover incase adjacent actors fail at the same time.
- If a critical actor ‘‘A’’ picks a non-critical neighbor ‘‘B’’ as a backup, RAM requires ‘‘B’’ to also pick a backup ‘‘C’’ among its neighbors using the same criteria mentioned above. However, node ‘‘B’’ status is not changed to critical. This condition enables recovery when the primary and backup both fail at the same time. In addition, it prevents the scenario of Fig. 4.

4.3. Failure detection and recovery

The local selection enables DCR to be applied in a distributed manner and scale for large networks. Figure 5(a) shows the setup where critical actors appoint their backups. The arrow head point towards the primary. Note that DCR does not require extra actors for serving as backup. It employs existing actors just to take care of each other.

Once an actor receives a BACKUP notification, it starts monitoring the primary through HEARTBEAT messages. The failure of the primary is detected by corresponding backup through successive misses of HEARTBEATS. Figure 5(b) indicates that the backup node B detects the failure of primary F and triggers the recovery process as detailed in the following section.

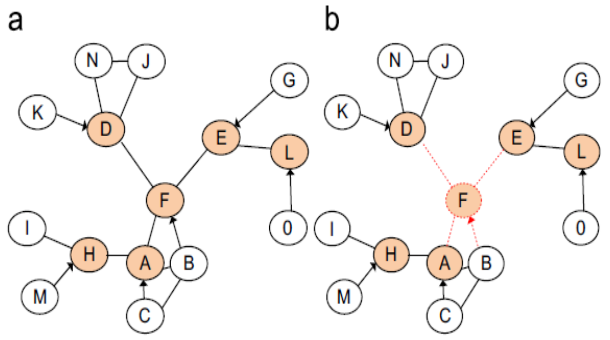


Fig 5:Critical actors designate their backup using DCR for network segment (a) backups start monitoring their primary and (b) B detects failure of primary F.

Like DCR, actors periodically exchange heartbeat messages with neighbors in order to update their status. The backup actor detects the failure of primary through missing successive heart-beats. Once the failure is confirmed, the backup node (s) initiates a recovery process that depends on the NCA. Since, our backup selection criteria strive to ensure that critical actors have distinct backups, the recovery procedure is executed concurrently on the various backups.

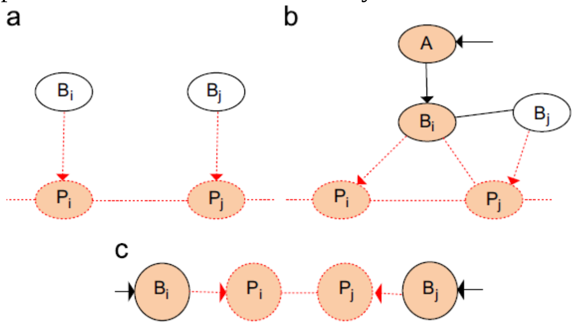


Fig 6: Recovery process when there is no risk in repartitioning the network and when the backups are (a) both noncritical and (b) one critical and one noncritical (c) both Critical.

If the backups are non-critical nodes, they simply replace the corresponding primaries and the recovery is complete. Figure6(a) demonstrates that replacing non-critical B_i and B_j restores the connectivity lost due to failure of P_i and P_j and does not require cascaded relocations. On the other hand, if the backup is a critical actor, moving that node will further trigger cascaded relocations until a non-critical node is engaged. For instance, Figure 6(b) demonstrates the scenario where a critical backup B_i sends a movement notification message to its own backup and moves to the place of failed primary P_i. Moving critical node B_i initiates a shifted relocation [15] where each backup replaces its corresponding primary. Whereas, a non-critical backup B_j simply moves to the location of the primary P_j

and the recovery is complete. Figure 6(c) shows the recovery process when both B_i and B_j are critical nodes. They will send a message to their backups and replace P_i and P_j, respectively.

Failure of adjacent actors: The presented recovery process of RAM will successfully restore the connectivity except for the following case for which the topology may not get repaired. A critical actor A_i may choose an adjacent critical node A_j as backup, while A_j designates another node A_k as a backup and A_k happens to be a neighbor of A_i. RAM can partially recover from failure of adjacent critical actors since one of the designated backup is also failed and none of the other nodes is responsible for the recovery. An example is depicted in
Figure 7(a)

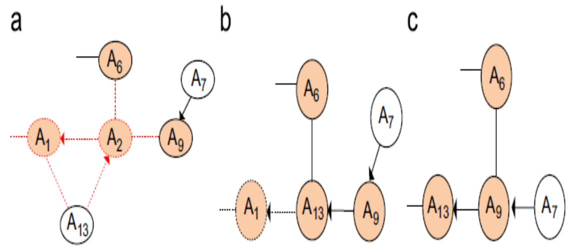


Fig 7:Special case of failure detection and recovery (a) A_{13} detects the failure of A_1 and A_2.
(b) A_{13}replaces A_2 and appoints A_9 as backup and (c) A_{13} moves to the place of A_1, A_9 and A_7follow it

Actor A_2 was a backup of A_1 and A_{13} was a backup of A_2. In case both A_1 and A_2 fail, although the backup A_{13} detects the failure of A_2 and executes the recovery procedure described earlier, none of the surviving nodes is responsible for tolerating the failure of A_1. Figure 7(b) clearly indicates that RAM cannot restore connectivity although the failure of A_1 was detected by neighbors. The obvious reason is that for particular critical actor RAM designates a backup that is only responsible for replacing such a critical actor when it fails.

To handle this case, we introduce a variant of RAM's recovery procedure that imposes slightly extra recovery overhead. The idea is to let backup know about the grand primary as well (i.e. primary of primary). In case of failure of adjacent critical actors, the designated backup coordinates the recovery. For instance, in Figure 7(a), A_2 makes A_{13} aware that it is a backup for A_1. In case of A_1 and A_2 fail, A_{13} will replace A_2 and find that A_1 is also lost as shown Figure 7(b). A_{13} appoints a new backup i.e. A_9, sends notification message and moves to replace A_1.

The newly appointed backup follows the primary and cascaded relocations are performed as shown in Figure 7 (c). This special case can be generalized to a ring of critical nodes in which A_2 serves as a backup to A_1, A_3 is backup of A_2, y, A_n is a backup of A_{n_1}, and, A_n is a neighbor of A_1.Since nodes A_1, A_2, y, and A_{n_1} are critical, if they fail, A_n needs to replace A_{n_1}. RAM will recursively send the primary-backup relationship of a series of reachable critical nodes on the ring. This can simply be achieved by making a primary C to inform its backup node B about whether C also serves as a backup for another primary A. If B has a link to A, B will apply this procedure. Otherwise if B is a critical node, A will keep on informing its backup about B and C and so on.

Figure 8(a) illustrates a slightly different scenario. A_3detects the failure of non-critical primary A_4 and finds that A_5 (grand primary) has also failed. A_3 ignores the failure of A_2 (since it is non-critical) and moves to the position of A_5 as shown in Figure 8 (b).

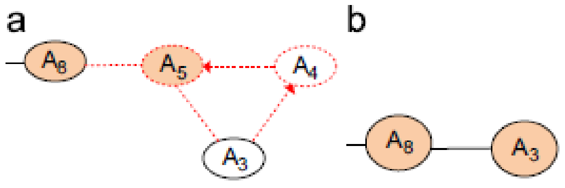


Fig 8:Special case of failure detection and recovery (a) A_3detect failure of critical actor A_5 and non critical actor A_4 and(b) A_3directly replace A_5 and ignores the A_4

V. LEAST-DISRUPTIVE TOPOLOGY REPAIR

The goal for LeDiR is to restore connectivity without extending the length of the shortest path among nodes compared to the prefailure topology.

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.

The following highlights the major steps in LeDiR.

(i)Failure detection: Once a failure is detected in the neighborhood, the one-hop neighbors of the failed actor would determine whether the failed node is critical node or not using the SRT.

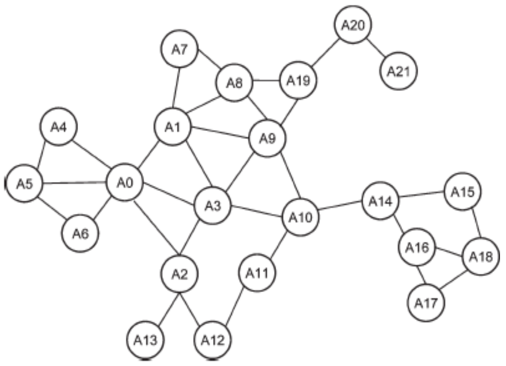


Fig 9:Example for connected inter actor network

(ii) *Smallest block identification*: 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.

(iii)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.

(iv)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.

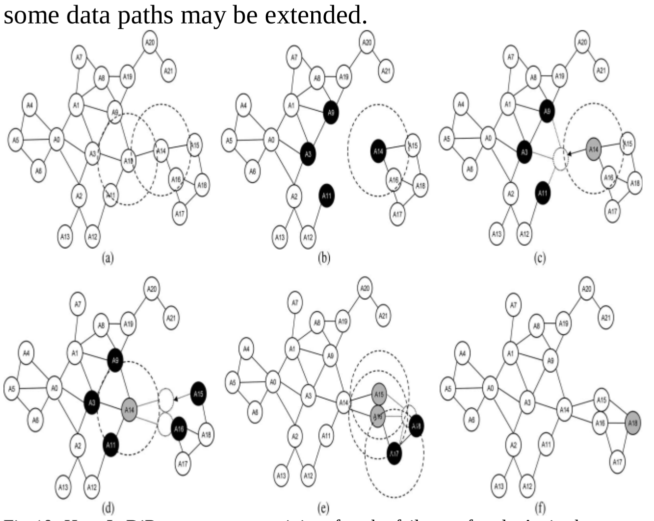


Fig 10: How LeDiR restores connectivity after the failures of node A_{10} in the connected inter-actor topology of Fig:9. Nodes in black are participating in the recovery process and those in gray are selected to move

Fig. 10 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 one-hop neighbor that belongs to the smallest block [see Fig. 10(a)–(c)]. In Fig. 10(d), node A_{14} notifies its neighbors and moves to the position of A_{10} to restore connectivity. Disconnected children, i.e., nodes A_{15} and A_{16}, follow through to maintain communication link with A_{14} [see Fig. 10(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 [see Fig. 10(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. Fig. 10(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.

VI. ALGORITHM FOR RAM & LeDiR

6.1 Pseudo-Code for RAM

The pseudo code of RAM shows that each actor ‘‘B’’ would execute the pre-failure steps resemble DCR. During the network bootstrapping phase, each actor (either critical or engaged as backup) will appoint an appropriate backup among neighbor actors using the AppointDistinctBackup ( ) procedure (lines 1–3). If actor B either detects the failure of primary F or receives a movement notification message from F, node B triggers a recovery procedure Failure Recovery () to recover from F (lines 4–6).

The AppointDistinctBackup () procedure is slightly different from its counterpart ‘‘Assign Backup’’ in DCR. The Appoint Distinct-Backup () procedure ensures that the picked backup node does not serve another primary and bases the selection on the criteria. The procedure Failure Recovery () is also different from the ‘‘Recovery’’ in DCR since in RAM two adjacent actors are not allowed to choose each other as backup at the same time. If the backup B is a critical actor, it notifies its backup so that the connectivity can be maintained (lines 8–10). Since backup B is aware of the status of the failed primary F, it checks whether the failed primary was critical or not. If the failed node F was critical B moves to replace F (lines 11–13). Otherwise, no need to replace since it was non-critical. In other words, B will directly move to the location of grand primary Gas.

If the backup node B also detects the failure of its grand primary G (i.e. primary of primary) then B executes the recovery procedure Failure Recovery ()to recover from grand primary (lines 14–16).

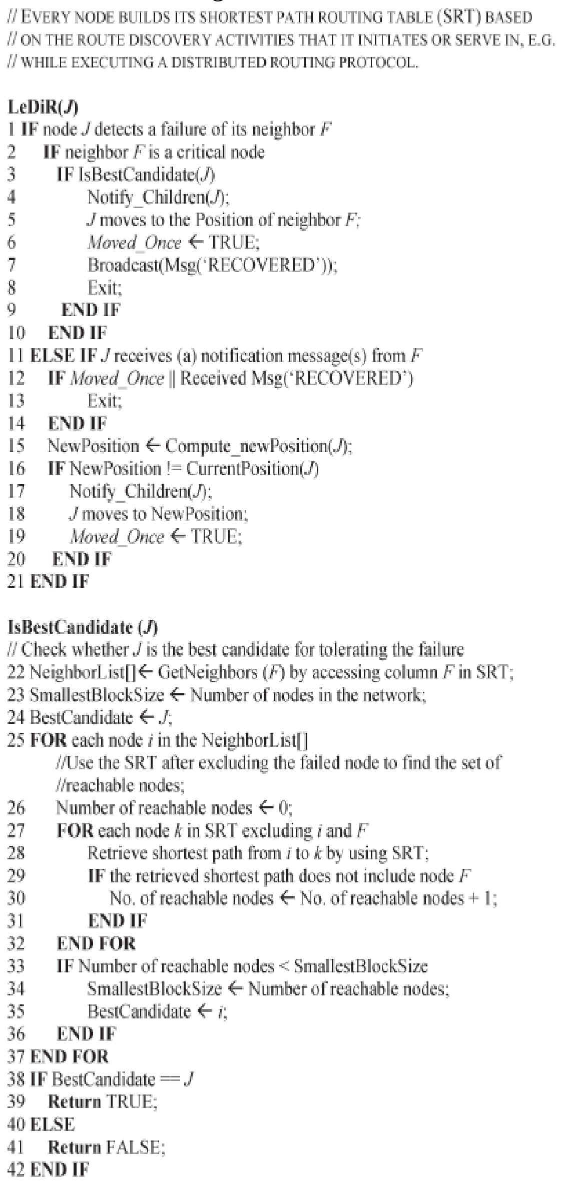
```
RAM (B)
1  if (IsCritical(B) == true || B.State == Engaged)then
2     AppointDistinctBackup (B)
3  end if
4  if (B detects the failure of primary actor F or receives a
   Movement notification message from F) then
5     FailureRecovery (B, F)
6  end if
AppointBackUp (B)
7  //Appoint a distinct backup for actor B based on the
     Criteria mentioned in Section-IV(C).
FailureRecovery(B, F)
8  if (IsCritical (B) == true) then
9     NotifyBackUp(B)
10 end if
11 if (IsCritical (F)) then
12    MoveToLocation(B, F)
13 end if
14 if (B detects failure of its primary's primary G) then
15    FailureRecovery (B, G)
16 end if
```

6.2 Pseudo-Code for LeDiR

The pseudo code for LeDiR is executed independently by each neighbor *J* of the failed node *F*. When an actor *J* detects the failure of a neighbor *F* , it applies depth-first search to determine that *F* is indeed a critical node (lines 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* qualifies, it will 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 (lines 12–14). The *Compute_newPosition*(*J*) procedure identifies where a node *k* (a child of *J*) would need to reposition based on the notifications that it has received from nodes other than *J*. The details of the *Compute_newP osition*(*J*) procedure and corresponding analysis

A new position for a node *k* would be computed only if *k* loses its direct communication link to one or multiple parent neighbors, under child movement. The *IsBestCandidate*(*J*) procedure identifies the smallest block using the SRT, as described earlier in Section IV, and whether the node *J* belongs to that block



```
// 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
```

VI. 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 with-out extending the length of data paths. The proposed algorithm identifies critical actors in advance based on localized information and designates for them backup actors. In order to handle multiple simultaneous failures of critical actors, we have proposed RAM. RAM handles failure scenarios in which two adjacent nodes simultaneously fail. Like DCR, RAM is also a distributed hybrid approach that identifies critical actors and assigns for them backups. However, RAM assigns distinct backups for each critical actor. The designated backups detect failure of their primaries and move to replace them. In addition, RAM extends LeDiR for restoring the network connectivity with minimal topological changes. In the future, we plan to evaluate the performance of the proposed approaches in a prototype network of mobile robots.

REFERENCES

[1] I. F. Akyildiz and I. H. Kasimoglu, "Wireless sensor and actor networks: Research challenges," *Ad Hoc Netw. J.*, vol. 2, no. 4, pp. 351–367, Oct. 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 al-gorithm for networks of moveable sensor nodes," in Proc. IEEE GLOBE-COM, 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] Abbasi, AA, Akkaya, K, Younis, M. A distributed connectivity restoration algorithm in wireless sensor and actor networks. In: Proceedings of the 32nd IEEE conference on local computer networks (LCN 2007), Dublin, Ireland; October 2007.

[8] Akkaya, K, Thimmapuram, A, Senel, F, Uludag, S. Distributed recovery of actor failures in wireless sensor and actor networks. In Proceedings of the IEEE wireless communications and networking conference (WCNC 2008), Las Vegas, NV; March 2008.

[9] Azadeh, Z. A hybrid approach to actor _actor connectivity restoration in wireless sensor and actor networks. In: Proceedings of the 8th IEEE international conference on networks (ICN 2009), Cancun, Mexico; March 2009.

[10] Younis M, Lee S, Abbasi AA. A localized algorithm for restoring inter-node connectivity in networks of moveable sensors. IEEE Transactions on Compu-ters 2010;99(12).

[11] Tamboli, N and Younis, M. Coverage-aware connectivity

restoration in mobile sensor networks. In: Proceedings of the IEEE international conference on communications (ICC 2009), Dresden, Germany; June 2009.

[12] Imran, M, Younis, M, Said, AM, Hasbullah, H. Volunteer-instigated connectivity restoration algorithm for wireless sensor and actor networks. In: Proceedings of the IEEE International Conference onWireless Communications, Networking and Information Security (WCNIS 2010), Beijing, China; June 2010.

[13] Akkaya K, Senel F, Thimmapuram A, Uludag S. Distributed recovery from network partitioning in movable sensor/actor networks via controlled mobility. IEEE Transactions on Computers 2010;59(2):258–71.

[14] MilenkoJorgic´, IS, Hauspie, Michael,¨ Simplot-ryl, David. Localized algorithms for detection of critical nodes and links for connectivity in ad hoc networks. In: Proceedings of the 3rd annual IFIP mediterranean ad hoc networking work-shop, Med-Hoc-Net, Bodrum, Turkey; June 2004.

[15] Li, N Xu S, Stojmenovic, Ivan. Mesh-based sensor relocation for coverage main-tenance in mobile sensor networks. In: Proceedings of the 4th international conference on ubiquitous intelligence and computing (UIC 2007), Hong Kong, China; July 2007.

[16] Ameer A. Abbasi, Mohamed F. Younis, Senior Member, IEEE, and Uthman A. Baroudi,"Recovering From a Node Failure in Wireless Sensor-Actor Networks With Minimal Topology Changes". IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, VOL. 62, NO. 1, JANUARY 2013,256-271

[17] Muhammad Imran a,n, Mohamed Younis b, Abas Md Said c, Halabi Hasbullah c , "Localized motion-based connectivity restoration algorithms for wireless sensor and actor networks", Journal of Network and Computer Applications 35 (2012) 844–856