Kuwait J.Sci., Vol.48, No.(4),October.2021,pp(1-16)

Distributed partition detection and recovery using UAV in wireless sensor and actor networks

Aditi Zear, Virender Ranga*
Dept. of Computer Engineering
NIT Kurukshetra, India
**Corresponding author: virender.ranga@nitkkr.ac.in*

Abstract

Wireless Sensor and Actor Networks (WSANs) have been extensively employed in various domains ranging from elementary data collection to real-time control and monitoring for critical applications. Network connectivity is a vital robustness measure for overall network performance. Different network functions such as routing, scheduling, and QoS provisioning depends on network connectivity. The failure of articulation points in the network disassociates the network into disjoint segments. We proposed Distributed Partition Detection and Recovery using Unmanned Aerial Vehicle (UAV) (DPDRU) algorithm, as an optimal solution to recover the partitioned network. It consists of three steps: Initialization, Operational and Detection, and Recovery. In the Initialization phase sink node collects all the information about the network. In the Operational and Detection phase, network nodes communicate regularly by exchanging HEARTBEATS, detects failure if some nodes do not get a message from the neighbor node and send failure reports, and sink node identifies network partition. In the recovery phase, the sink node sends UAV at the positional coordinates of the failed node and examines network recovery after UAV reaches the desired location. Our approach primarily focuses on reducing message overhead by sending few update messages to sink node and energy consumption by engaging network nodes only for communication. The requirements of the recovery process (physical movement and communication) are fulfilled by UAV. The algorithm is tested according to the following parameters: Detection Time, Recovery Time, message overhead, and distance traveled by UAV. Simulation results validate the efficacy of the proposed algorithm based on these parameters to provide reliable results. The minimum and the maximum number of messages transmitted are 11 for 10 nodes and 24 for 100 nodes respectively. Hence these results demonstrate that the message overhead in our proposed solution is less as compared to other techniques when the number of nodes increases.

Keywords: Knowledge table; partition detection; partition recovery; relay node; sensor network; Unmanned Aerial Vehicle.

1. Introduction

WSANs have been gaining the attention of the researcher's community because of their serviceability in rough environments. They reduce human engagement to attain intelligent and autonomous interactions with the environment (Younis *et al.*, 2014; Akkaya & Senel, 2009, Ranga *et al.*, 2013). WSANs comprise connected sensor and actor nodes through a wireless medium. Sensors report information related to multiple events to actors, and actors make quick decisions based on the received data and carry out actions. As compared to sensors, actors have better computation and communication capabilities. Actors can do sensing mechanisms. However, the decision-making capability is not present in sensors. The extensive applications of WSANs in different domains require effectiveness, reliability, and some degree of robustness (Bayrakdar, 2020**a**,**b**). However, the robustness of the network to handle different

kinds of failures depends on the specificity of the deployed nodes (Shriwastav & Ghose, 2018; Verma & Ranga, 2018; Bayrakdar, 2020**c**). The minimal characteristics of sensor nodes and their deployment in the rough environment result in a drastic decrease in the reliability of WSANs. Also, the interferences caused by the sensor and actor nodes increase collisions and the process of re-transmissions, which causes the premature failure of sensor nodes because of the increase in energy consumption. These factors affect the proper functioning of the network and lead to the failure of nodes without completing their actual mission or task. The failures can also create a network partition where parts of the deployed network get entirely disconnected from the base station (sink) node. Thus network connectivity to avoid such partitioning can be assured by integrating node deployment mechanisms along with recovery mechanisms (Stojmenovic *et al.*, 2011; Mahmood *et al.*, 2018; Lee *et al.*, 2015; Jha *et al.*, 2019; Senturk *et al.*, 2014; Joshi & Younis; 2016; Ranga *et al.*, 2016**b**; Lalouani *et al.*, 2017; Bayrakdar, 2020**d**).

The network partitioning can be either due to single cut-vertex failure or because of the simultaneous failure of multiple nodes that can include cut-vertex or non-cut vertices (Stojmenovic *et al.*, 2011). Restoring connectivity after failure because of unknown reasons with limited resources for the continuation of sensing and acting services is critical. The mechanisms to recover from simultaneous multiple node failures can be categorized depending on whether the damaged nodes are collocated or dispersed. The dispersed damaged node's failure recovery is similar to handling separate single-node failures when recovered in a centralized manner. On the other hand, autonomous recovery through distributed solutions is problematic due to resource conflicts. An example of such conflict is simultaneous requests to a node to assist recovery from two different individual node failures. Synchronization mechanisms are required to mitigate such conflicts. The most challenging scenario is mitigating the simultaneous failure of numerous collocated nodes because it becomes difficult to resolve the scope of failure (Shriwastav & Ghose, 2018; Senturk *et al.*, 2014; Abbasi *et al.*, 2010; Zhang *et al.*, 2018; Chouikhi *et al.*, 2017; Ma *et al.*, 2016; Hashim *et al.*, 2016; Joshi & Younis, 2014; Akkaya & Senel, 2009; Afzaal *et al.*, 2017; Senturk *et al.*, 2012; Akkaya *et al.*, 2010; Srinivasaperumal *et al.*, 2017; Akkaya & Senel, 2009; Bayrakdar, 2020**e**; Ranga *et al.*, 2016**a**).

The way of illustrating the deployed network using a graph is regarded as an efficient way for failure recovery. The failure recovery techniques are categorized into four types: Deploying redundant nodes, Node relocation, employing data mules to relay data (data ferrying), and placement of relay nodes. The first category is based on provisioning additional resources for establishing k-connectivity, with the aim that network operates even after some node damages (Akkaya *et al.*, 2010). This technique results in huge communication overhead and significant deployment expenditure. The second technique is based on the assumption that nodes are mobile and can move in a particular direction to restore communication links by relocating a subset of surviving nodes for failure recovery (Younis *et al.*, 2014; Shriwastav & Ghose, 2018; Chouikhi *et al.*, 2017). This solution requires complex hardware for network nodes and leads to high energy consumption (Younis *et al.*, 2014; Mahmood *et al.*, 2018; Lee *et al.*, 2015). The third technique is a temporary recovery mechanism where mobile data collectors (data mules) are used to transfer information between the partitioned clusters. This solution is not efficient because of the increased overhead and data latency. The utmost important fourth network recovery technique is through relay node deployment. It is the most popular and efficient mechanism where existing and additional nodes are deployed to re-establish connection among nodes (Jha *et al.*, 2019; Joshi & Younis, 2016; Ranga *et al.*, 2016**b**; Chouikhi *et al.*, 2017). Finding an optimal relay count required to restore links is an optimization problem and requires complete information about the number of partitions and locations of nodes in them (Younis *et al.*, 2014; Shriwastav & Ghose, 2018; Senturk *et al.*, 2014; Chouikhi *et al.*, 2017; Alfadhly *et al.*, 2011; Cheng *et al.*, 2017; Nikolov & Haas, 2016; Devi & Manickam, 2014; Zahid *et al.*, 2018; Liu *et al.*, 2018; Lee *et al.*, 2016; Kumar & Amgoth, 2019; Lalouani *et al.*, 2017; Ranga *et al.*, 2016**a**).

All the techniques mentioned above for network partition recovery are based on the assumption that nodes are moving in an obstacle-free environment, which is impossible in real environments. Various physical obstacles like buildings or mountains

obstruct the network communication. Dynamic obstacles such as a group of birds or animals and environmental conditions harm sensor network and blocks the connection. The less applicability of proposed solutions in practical situations can be solved by employing drones or Unmanned Aerial Vehicles (UAVs). UAVs are suitable because of their serviceability in areas that are not in the reach of ground nodes (vehicles or robots) due to obstacles and capability to work in harsh environmental conditions. These advantages of UAVs have increased their usage in several applications such as agriculture, disaster recovery, industrial inspections, military, and remote sensing. Therefore the objective of our research is to examine the performance of UAVs for network partition recovery (Zhang & Liu, 2019; Chriki *et al.*, 2019). Thus our research in this paper is based on distributed monitoring of neighboring nodes and reporting their failures to sink node. If the sink node addresses this failure as a network partition, then it employs UAV for the network recovery, as shown in Figure1.

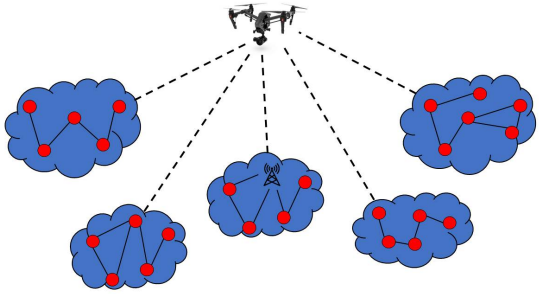

Fig. 1. UAV Connecting Disjoint Network Partitions.

In this paper, we proposed Distributed Partition Detection and Recovery using UAV (DPDRU). The main advantage of our approach is that it reduces the message overhead on the sink node, and uses UAV as the relay node for re-connection that can even work in rough environments with obstacles where normal network nodes cannot work properly. The approach that involves node re-locations dissipates energy for both node movements and communication. Therefore in our approach, network nodes are free from this overhead. The sink node and network nodes consume energy only for communication. UAV with better communication range, durable battery as compared to sensor nodes, is utilized for both movements and communication purposes. The algorithm is divided into three phases: Initialization, Operational and Detection phase, and Recovery phase. The initialization phase is used to collect all the network information by the nodes for the sink node. Depending on the collected information, the sink determines the cut-vertices in the network graph and stores them. The Operational and Detection phase is used to detect the failures and notifying the sink node about them. In the Recovery phase, UAV reconnects the partitioned network, and the sink node confirms that partitions are reconnected. Hence our proposed algorithm aims at detecting and recovering the network partition. The previously proposed solutions are less applicable in practical situations, and this issue can be solved by employing drones or Unmanned Aerial Vehicles (UAVs) that can even work in harsh environmental conditions. Hence instead of focusing on node relocation-based solutions, We propose to use UAV for re-establishing connectivity in the network partitions. Along with this, we have also tried to reduce the number of messages transmitted to detect and recover the network partitions.

The brief introduction of the remaining sections in the paper is depicted as follows: Section 2 provides related work of network partition detection and recovery solutions. The system model is depicted in Section 3, and Section 4 explains the phases of the algorithm along with pseudocode. Section 5 and Section 6 discuss the algorithm analysis of DPDRU and simulation results.

2. Background

Maintaining connectivity is an essential requirement of WSANs. Various efforts have been made to remodel network partitions to form connected sensor and actor-network topology. The proposed solutions can be divided into two categories: preventive and curative. The curative solution for failure management is triggered after the failure is identified. It includes different techniques, such as the deployment of additional nodes and reorganizing the partitioned network. The preventive approach for fault tolerance aims at handling network partitions before failure include avoiding damages of the node, and transmitting information through alternate paths, if some node fails (Younis *et al.*, 2014; Akkaya & Senel, 2009; Shriwastav & Ghose, 2018). These solutions focus on deployment strategies (relay nodes, redundant

nodes, etc.), multiple-path routing strategies, managing energy consumption (sensor activities, sleep schedule, multi-channel communication, etc.) (Chouikhi *et al.*, 2017; Kumar & Amgoth, 2019).

Partition Detection and Recovery Algorithm (PADRA) in (Akkaya & Senel, 2009) detects the partitioned segments in the network and recover the connections through the regulated displacement of sensor nodes. The partition in the network is identified by monitoring the failure of a cut-vertex. (Abbasi *et al.*, 2010) proposed the least disruptive topology repair algorithm (LeDiR), which relocates the block containing a smaller number of nodes among the network segments for recovery. It ensures that path length between the pair of network nodes does not increase as compared to the previously connected network. Distributed Node Recovery Algorithm (DARA) is proposed in (Abbasi *et al.*, 2007) to recover the disconnected network. It uses the least number of actor nodes for establishing connectivity. The Distributed Prioritized Connectivity Restoration Algorithm (DPCRA) (Ranga *et al.*, 2014) approach restores connectivity with minimum nodes. This hybrid approach assigns failure handlers to each cut-vertex nodes. When the network gets partitioned, these failure handlers perform the recovery process. In (Chouikhi *et al.*, 2017) two centralized approaches have been proposed for reconnecting disjoint segments resulted because of the failed cut-vertex node. The PFR algorithm takes preventive steps for making network robust so that network partition should not occur, and the second RNFR algorithm intent to recover the partitioned network.

Two-connected topology is formed in (Hwang *et al.*, 2014) to federate the disjoint segments of the network. At the first stage, the smallest convex hull is located, and each segment places the network's center point and then relays in the convex hell for connecting different segments. Federating network segments via triangular Steiner tree approximation (FeSTA) in (Senel & Younis, 2011**a**) represents the disjoint segment using terminal nodes, and a triangle is used to describe the subset of three terminals. The triangle is steinerized by reducing the edges of MST. This step either creates a new connected component or joins the terminal with already present connected components. The algorithm works quite well when tested according to network coverage, average path length, and average node degree. In (Lalouani *et al.*, 2017) Boundary-aware optimized Interconnection of Disjoint segments (BIND) algorithm uses the shorter length topology in the two-dimensional plane to restore the network. The subset of nodes is selected from the boundaries of disjoint segments and then interconnected using the additional Steiner pairs to form interconnected topology between the segments. In (Joshi & Younis, 2016), the Geometric Skeleton based Reconnection (GSR) approach shapes the partitioned network area in a distributed way to recover connectivity. The network area is transformed into the two-dimensional skeleton for placing relays along with this skeleton which acts as the support structure to reconnect the network

Cell-based Optimized Relay Placement (CORP) algorithm in (Lee & Younis, 2010**b**) divides the area into equal size cells or grids. The cells present at the minimum distance between two pairs of segments are called the best neighbor cell. The algorithm works in several iterations. Each round consists of electing the best cell, and relay nodes are populated in the selected cell. The process continues until the segments form connected topology by including relay nodes. Partition detection and Connectivity Restoration (PCR) algorithm in (Imran *et al.*, 2010) determined the critical-actor node from the local topological data. Actor nodes that function as backup monitor the critical nodes continuously. After detecting the failure, they coordinate their movements towards the failed critical node. These displacements are continued until the connectivity is re-established. The work in (Senel & Younis, 2011**b**) represents each partition using all the nodes present in the boundary. It constructs a minimum spanning tree between boundary nodes and computes the required relay count to set up the network with no partitions. The main objective is to reduce the total path length. Connectivity Restoration with Assured Fault Tolerance (CRAFT) algorithm in (Lee *et al.*, 2015) creates a larger inner cycle, which is the backbone polygon (BP). No partition resides in this backbone polygon as BP is constructed around the failure region. The relay nodes are placed on BP, and every outer partition is then connected to BP. The algorithm works by reducing the path length between each pair of segments, and network topology is made bi-connected by deploying relays

with a high node degree. (Lee & Younis, 2010**a**) proposed a Distributed algorithm for Optimized Relay placement using a Minimum Steiner tree (DORMS) algorithm in which relay nodes are reduced with the help of the Minimum Steiner tree. The relay nodes selected from the existent nodes in each partition are moved towards the middle of the deployed network until these nodes come in the communication radius of each other, and the disconnected partitions start operating again. The main objective of the presented approach is to plan an organized topology with the minimum number of additional relay nodes. However, the shifting of existing nodes also causes overhead. Optimized Bi-Connected federation of multiple sensor network segments (OBiC) in (Lee *et al.*, 2016) creates the Hamiltonian cycle such that every segment can be reached once. The algorithm works in two phases: The first phase consists of inward connecting of outer segments towards the center where each segment connects to the closest inner segment. On the other hand, the second phase forms the concave hull, and its circumference is minimized in the third phase to place the fewer relay nodes on every edge. The (Han *et al.*, 2009) considered the different communication radius of all deployed nodes in the network, which resulted in asymmetric links between the neighboring nodes. Therefore a systematic fault tolerance is required. An approximation algorithm is proposed to tolerate faults, and its performance is evaluated for various networks.

3. System model

We considered a randomly deployed network consisting of sensor and actor nodes. The base station or sink node is static, and it collects data from other deployed network nodes. The nodes are stationary. However, they can move when required. The Sink node is reachable to all network nodes through direct or multi-hop links. A direct link is present between the two if the distance between them is less than their communication radius. Nodes obtain information about their immediate one-hop neighbors by sending periodic BEACON or HELLO or HEARTBEAT messages. All nodes are aware of their location and communicate their position coordinates to the sink node when the network is initialized. The network can contain articulation (cut-vertex) and non-articulation nodes (non-cut-vertices). Various factors such as electrical and radio frequency interference, environmental factors, physical objects affect the deployed wireless nodes. If there are missing HEARTBEAT or HELLO messages, then the node is considered as failed by its neighbors. The failure of the cut-vertex disconnects the network into multiple partitions. In such scenarios, the sink node does not get information from the nodes in other partitions. Therefore it takes a decision and performs recovery operation based on important information it receives from the surviving nodes. The network is depicted as a graph G (V_i, E_i) where V_i represents the nodes (actors or sensors) in the network, and E_i depicts edges between the nodes which are in each other's communication radius. This system model consists of the following assumptions:

1. All the nodes and UAVs have the same communication range.
2. Single cut-vertex-node fails at a time, and during recovery, there will be no damage in other network nodes.
3. Nodes are aware of their locations.
4. All network nodes have finite energy and cannot be recharged except the sink node.
5. The sink node does not fail.

4. Proposed approach

The proposed DPDRU algorithm consists of Initialization Phase, Operational and Detection phase, and Recovery phase, which are described below:

4.1 Initialization phase

Network configuration starts when each node starts transmitting a HELLO message to its 1-hop neighbors. When a node receives a HELLO message from a node, it includes it into its 1-hop neighbors (OHN) list. After this step, the nodes send their node_ID, position coordinates (x_{coord}, y_{coord}), and the newly created one-hop neighbor list to the central sink node. Sink node constructs Knowledge Table (KT) after receiving data from all nodes. The entries of KT are node_ID and their position coordinates (x_{coord}, y_{coord}). The sink also constructs the adjacency list (Adj) that contains the one-hop neighbors (OHN) of all nodes in the network. The example network in Figure 2 shows

its corresponding Adj in Figure 3. Sink node then executes Depth First Search (DFS) algorithm on this adjacency list to find out the cut-vertices or articulation points in the graph whose failure can cause network partitioning. Nodes in the DFS tree are cut-vertices, if they meet these two properties given below:

1. Root node with at least two child nodes is cut-vertex.
2. The internal node (N) of the DFS tree is cut-vertex if at least one sub-tree rooted at one of the child nodes of N should not have an edge connected to any ancestors of node N.

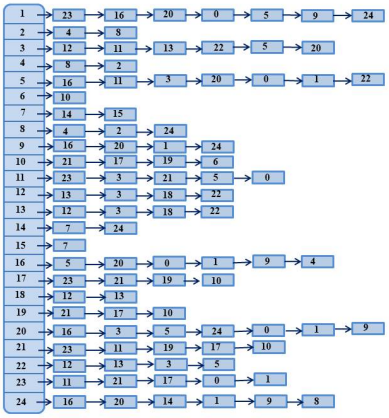

Fig. 3. Adjacency list (Adj) of Example Network

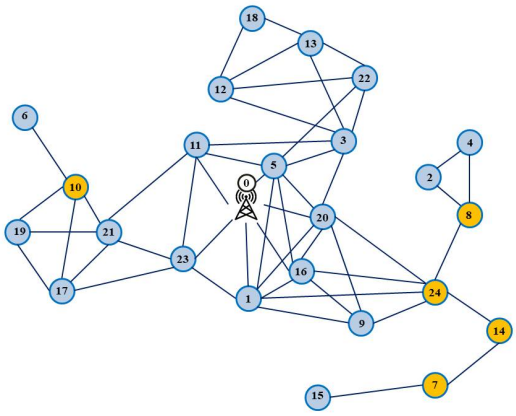

Fig. 2. Example Network

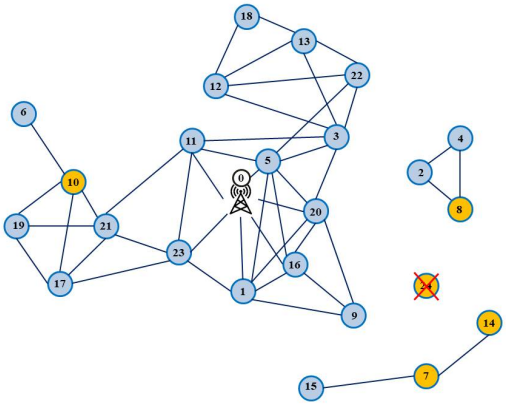

Fig. 4. Partitioned Network

The output of the DFS algorithm consisting of cut-vertices or articulation points of the given network is stored in a separate array called CUT_VERTICES. Referring to Figure 2, the cut vertices in the example network are 10, 8, 24, 14, 7. The pseudocode of this figure is given in Figure5. Line 1-5 explains sending the HELLO packet, receiving packets from neighbors, and creating the one-hop neighbor list.

4.2 Operational and detection phase

In our proposed approach, total messages exchanged among the sink node and the network nodes are reduced as compared with state-of-the-art approaches. The network works according to the standard procedure. The nodes send HEARTBEATs to their neighbors (OHN) after some pre-determined time interval. This time interval varies according to the type of application. However, the UPDATE messages that nodes send to inform the sink node about several events are reduced. In our proposed approach, network nodes send UPDATE messages to the sink node only to notify about important events. These essential updates include sending the one-hop neighbor list at the initial stage, failure, and recovery reports.
When a node in the deployed network does not receive two HEARTBEAT messages from its neighboring node (1-hop neighbor), then it considers it failed node. After identifying the failure of the adjacent node, it sends a failure report containing the node_ID of the failed node to sink. The sink node compares the received node_ID of the damaged node with the CUT_VERTICES. If the failed node is one of the cut-vertices, then the sink acknowledges this failure as a network partition. It stores the node_ID of the neighbor node, which reported the failure of damaged cut-vertex in a separate array called FRNN (Failure Reporting Neighboring Nodes).

After acknowledging network partition, the sink

```
INITIALIZATION
PROCEDURE AT NETWORK NODES
1.  for i = 1 to N
2.     Send HELLO messages to nodes in communication range.
3.     Construct OHN list after receiving HELLO from neighbor nodes.
4.     Send one-hop neighbor list and location information to sink.
5.  end for

PROCEDURE AT SINK NODE
1.  for i = 1 to N
2.     Receive UPDATE messages from N network nodes (node_i).
3.  end for
4.  Construct Knowledge Table (KT).
5.  Construct Adjacency list Adj.
6.  Apply DFS on Adj to identify cut-vertices and non-cut vertices.
7.  Constructs CUT_VERTICES array consists of cut-vertices.
```

Fig. 5. Initialization Phase

node enforces the partition recovery procedure. Referring to Figure 4, when node 24 fails, the resulting network consists of three disjoint partitions. The one-hop neighbors of 24 are 16, 20, 14, 1, 9, and 8. The sink does not receive a failure report from 14 and 8, because they are not present in other partitions. Therefore, the nodes 16, 1, 9, and 20 will be recorded in FRNN. The pseudocode in Figure 6 explains the working of this phase. Lines 1-6, in 'PROCEDURE AT NETWORK NODES', describes the detection process to identify the failed node and transmit its information to the sink. Lines 1-6 in 'PROCEDURE AT SINK NODE,' sink node identifies network partition by comparing failed node ID with CUT_VERTICES. The decision related to handling network partition by calling 'RECOVERY PROCEDURE AT SINK NODE' or no recovery required is made in lines 7-12. The ID of the failure reporting node is stored in FRNN.

4.3 Recovery phase

In the recovery phase, UAV is employed to place the stationary node to establish the connection. The Sink node sends UAV to the positional coordinates of failed cut-vertex. Upon reaching the desired locations, UAV deploys the nodes, and deployed nodes send HELLO messages to the nearby nodes that lie in its transmission radius, i.e.,1-hop neighbors. Network nodes add node_ID to their 1-hop neighbor list after receiving the HELLO message from the deployed node. After including deployed node in the list, these nodes send a recovery report to the sink node containing this updated list. Sink node records the node_ID of nodes submitting recovery report in RRNN (Recovery Reporting Neighboring Nodes) array.

After that, the sink compares the elements of FRNN with all the entries in RRNN. If all the node IDs in FRNN will be present in RRNN, then the sink node considers it as partition recovered. It empties FRNN, RRNN, and updates the Adjacency matrix (Adj) with new one-hop lists of the nodes.

Referring to Figure 7, UAV reaches the location of 24 and deploys a recovery node at that location. Then deployed node sends a HELLO to the neighboring nodes in its communication range. Upon receiving the HELLO message, the nodes that had detected the failure, i.e., 16, 1, 9, 20, 14, and 8, update their one-hop list. These nodes then send the recovery report to the sink, and the sink records their node ID in RRNN. The number of nodes in RRNN can be higher than the node entries in FRNN, because the sink node can now receive messages from all the nodes that had detected the partition. The sink node terminates the recovery procedure when the node IDs in FRNN, i.e., 16, 1, 9, and 20 are entirely present in RRNN. This step implies that UAV has established the lost connectivity again, and all the nodes that had sent the failure report to the sink node previously have now sent a recovery report. The pseudocode in

```
OPERATIONAL AND DETECTION PHASE
PROCEDURE AT NETWORK NODES
1.  for i = 1 to N
2.     Receive HEARTBEAT messages from node_i.
3.     if (missing HEARTBEAT message from node_i)
4.        failed_node = node_i
5.        Send failure report including failed_node to sink node.
6.     end if
7.  end for
PROCEDURE AT SINK NODE
1.  Receive failure report from network nodes
2.  for i = 1 to V where V is no of cut vertices
3.     if (failed_node = CUT_VERTICES[i])
4.        failed_cut_vertex = failed_node
5.        Acknowledge Network Partitioning
6.        Store failure reporting node into FRNN
7.        Call RECOVERY PROCEDURE AT SINK NODE
8.     end if
9.     else
10.       Recovery procedure not required
11. end for
```

Fig. 6. Operational and Detection Phase

Figure 8 explains the recovery procedure. Lines 1-7 demonstrate the working of the sink node sending UAV to the location of failed cut-vertex and waiting for UPDATE message from failure reporting nodes. If the sink receives a recovery report, then it stores the recovery reporting node's ID in RRNN and calls procedure Rec_conf (FRNN, RRNN) in line number 6. Line 8-23 explains the process of comparison between FRNN and RRNN to check whether FRNN is entirely present in RRNN. Based on the results, it decides whether the network is recovered or not. It updates Adj if the network is recovered in line number 20.

5. Algorithm analysis of DPDRU

The brief algorithmic analysis of the detection and recovery processes of the proposed DPDRU algorithm is illustrated below:

Lemma 1: *DPDRU connects partitions successfully and aborts successfully.*

Proof: The sink node keeps a record of the failure reporting nodes and stores their ID in a separate array. When these nodes receive HEARTBEAT messages from deployed UAV node, they include them in the OHN list and send a recovery report to the sink node. Sink stores the node IDs of recovery reporting nodes in a separate array and compares it with that failure reporting nodes array. If all failure reporting nodes are present in the recovery reporting array, then the sink confirms that UAV has recovered the network partition.

Lemma 2: *The utmost distance traveled by UAV during recovery in DPDRU is equivalent to* $\sqrt{\frac{d^2}{2}}$*, where* $d \times d$ *are the dimensions of the area where nodes are deployed.*

Proof: The UAV is present at the center of the network. In the worst case, when cut-vertex is at the corner of the region fails, then it moves from its location to that corner, and that distance is equivalent to half the length of the diagonal of squared area.

Length of diagonal = $\sqrt{d^2 + d^2} = \sqrt{2d^2}$

And distance traveled by UAV = Length of diagonal/2 = $\frac{\sqrt{2d^2}}{2} = \sqrt{\frac{2d^2}{4}} = \sqrt{\frac{d^2}{2}}$

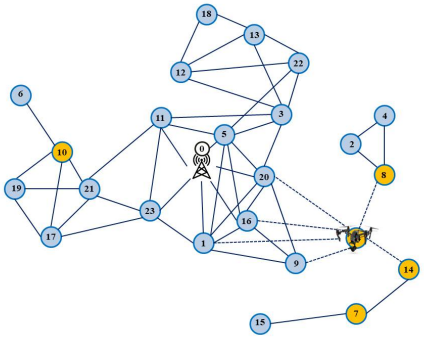

Fig. 7. UAV Restoring Network Partition

```
RECOVERY PHASE
RECOVERY PROCEDURE AT SINK NODE
1.  Derive coordinates (xf , yf) of failed_cut_vertex from KT
2.  Send UAV at the location (xf , yf)
3.  Wait for recovery UPDATE message
4.  if (recovery report from nodei)
5.     Store the node_id of reporting node in RRNN
6.     Call Rec_conf(FRNN, RRNN)
7.  end if
8.  Rec_conf()
9.  {
10. int count = 0
11. for i = 1 to FRNN.size()
12.    for j = 1 to RRNN.size()
13.       if(FRNN[i] == RRNN[j])
14.          count = count+1
15.       end if
16.       Break
17.    end for
18. end for
19. if (count == FRNN.size())
20.    Confirm Network Recovery
21.    Update Adj with new one-hop neighbor list of reporting nodes
22. end if
23. else
24.    Acknowledge Network is not Recovered
25. }
PROCEDURE AT FAILURE REPORTING NODES
1.  Receive HEARTBEAT messages from neighboring nodes
2.  if (HEARTBEAT from UAV recieved)
3.     update one-hop neighbor list
4.     send recovery report to sink along with updated OHN list
5.  end if
```

Fig. 8. Recovery Phase

Lemma 3: *DPDRU takes total (2(T_{delay} + T_{var})+ x/v) recovery time, where T_{delay} is the time interval between sending the heartbeat messages, T_{var} is the time between which HEARTBEAT messages are transmitted, x is the distance traveled by UAV with velocity v.*

Proof: The nodes in the network wait for the T_{delay} period before sending HEARTBEAT messages to the neighboring nodes, and they send between the time intervals T_{var}. DPDRU waits for two missing HEARTBEAT messages to detect the failed node. Therefore the average time required for the detection is 2(T_{delay} + T_{var}). After the failure detection, they immediately send a failure report to the sink. If the sink detects network partition, then it sends UAV at the location of failed cut-vertex to deploy a stationary node for recovery. Therefore, the time required for the recovery (NRT) directly depends on the distance traveled by UAV to reach the site of the failed node, and it is indirectly proportional to the velocity of UAV, i.e., NRT is proportional to x/v. Therefore the total time required by the algorithm for recovery after failure is equivalent to (2(T_{delay} + T_{var})+ x/v).

Lemma 4: *The best case and the worst case message complexity of DPDRU are O(1) and O(N), respectively.*

Proof: In the best case, the failed cut vertex has two 1-hop neighbors (OHN), and one of them is present in the sink node's partition. Therefore, sink receives a failure report from only one node, and when the network recovers, it gets two recovery nodes, which is equivalent to three messages. If each node sends recovery and failure reports twice, then in the best case, the total number of messages will be six, a constant number. Therefore the best case message complexity is O(1). In the worst case, the failed cut-vertex can have N-1 one-hop neighbors, where N is the total number of deployed network nodes, and N-2 nodes are present in the sink node's partition. If failure and recovery reports are sent twice, then the sink will receive 2(N-2) failure reports and 2(N-1) recovery reports. Therefore total messages are equivalent to 2((N-2) + (N-1)) = 4N-6. Thus, the worst case message complexity for the recovery process is O(N).

Table 1. Simulation parameters

Parameter	Value
Simulation Area	400m×400m - 1000m×1000m
Nodes	10 - 100
Transmission Range	80m - 200m
Packet Size	512bytes
Mobility model of nodes	On-demand mobility
Mobility model of UAV	Linear mobility
UAV speed	$2ms^{-1}$ - $30ms^{-1}$
UAV altitude	10m
Simulation Time	1000s
Data rate	54 Mbps

6. Simulation results and analysis

OMNeT++ 5.4.1 is used to implement and test the working of our proposed algorithm. The algorithm is implemented in the application layer of the INET 4.0 project. Table 1 describes the parameters used in the simulations. The algorithm is tested for different node topologies containing 10-100 nodes. These topologies include articulation points and deployed in an area ranging from 400m×400m to 1000m×1000m randomly. The communication range of deployed nodes varies between 80m-200m. UAV is present at the center of the area near the sink node, where all nodes are deployed.

Initially, all the nodes send update messages, including their one-hop neighbors and location information to the sink node. The Sink stores all this information and find out the cut-vertices in the graph. If nodes detect some failure, it sends the failure report (FR) to the sink node. The Sink node finds out whether it is a simple node failure or cut-vertex failure. If cut-vertex fails, it commands UAV to reach the location of the failed node to deploy the node for recovery. After getting a message from a deployed node, failure reporting nodes send recovery reports to the sink node. We have proposed a distributed failure detection algorithm and partition recovery using UAV. In the proposed algorithm, we have tried to reduce the messages overhead during the detection and the recovery process compared with state-of-the-art approaches. This message overhead is compared

with other similar approaches for network partition recovery from the single cut vertex failure such as LeDir (Abbasi *et al.*, 2010), DARA (Abbasi *et al.*, 2007), PADRA (Akkaya *et al.*, 2010), DPCRA (Ranga *et al.*, 2014). These approaches recover partitioned segments by relocating the nodes that survived after failure. Most of the parameters used for testing these approaches are based on this node relocation process such as total distance traveled by nodes, energy consumed due to relocation process, coverage reduction, messages transmitted, and the number of relocated nodes. Therefore, we performed the comparative analysis based on message overhead only. The recovery is performed with the help of UAV, which also reduces the recovery time. The details of parameters used for the analysis and the experimental results based on them are described below:

1. **Partition Detection Time (PDT):** Detecting the network partition is an essential step for the recovery. Many approaches in the literature assume the number of partitions formed after the failure and start the recovery process. Our proposed algorithm is not based on this assumption. Therefore we also tested the algorithm for the partition detection time (PDT), which is not present in other proposed solutions. It is the time interval between the failures of the cut-vertex node and sink determining network partition after receiving the failure report from other network nodes. Figure 9 refers to the results of PDT w.r.t number of nodes; it is almost the same for 10 and 20 nodes. After that, it increases with the growth in network size. However, some deviation is also present in results, e.g., detection time of 50 nodes and 100 nodes are less than the detection time of 40 nodes and 90 nodes, respectively. These deviations are because of various factors such as T_{var}, T_{delay}, and the total number of one-hop neighbors (OHN) of a failed cut-vertex. The increasing number of nodes increases T_{var} because of more variations in starting time when nodes start sending. The delay time T_{delay} varies from application to application. It should be less for real-time applications. If the node fails after sending the HEARTBEAT message, then its failure can be detected after waiting for the whole T_{delay} period. If the node fails before T_{delay} then its failure can be detected immediately by neighboring nodes. PDT is directly related to OHN because if one-hop neighbors of failed cut-vertex are more, then the sink node receives failure report from many nodes, which takes time because of the previously discussed parameters. Figure 10 describes the detection time performance of the proposed algorithm for T_{delay}. The detection time grows with an increase in the value of T_{delay}. There are some slight variations in the graph where detection time reduces with an increase in T_{delay}. These variations depend on the time when the node is failed, i.e., before T_{delay} or after T_{delay}.

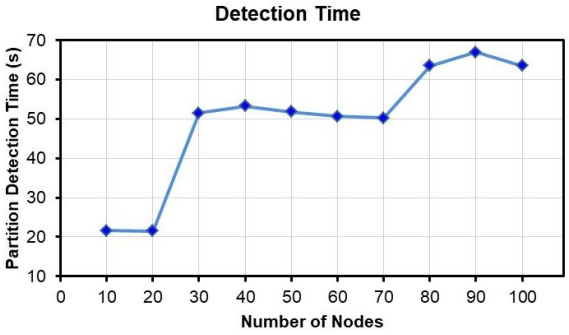

Fig. 9. PDT w.r.t number of nodes

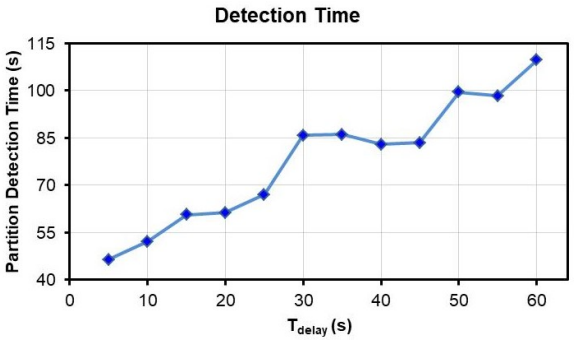

Fig. 10. PDT w.r.t T_{delay}

2. **Messages Transmitted during Recovery (MTR):** This parameter is used to analyze message overhead during the partition recovery process. In the proposed approach, we reduced the messages exchanged between the nodes and the sink. After two missed HEARTBEAT messages, nodes send the failure report to the sink node. The Sink after receiving these updates, examines the network condition. In the case of partitioning, it commands UAV to reach the location of the failed node. When UAV reaches that location,

it sends HEARTBEAT messages similar to the normal working of network nodes. When failure reporting nodes add UAV to their OHN list, they send a recovery report to the sink node. Figure 11 illustrates the comparative analysis of the proposed algorithm based on MTR with other similar approaches such as LeDir (Abbasi *et al.*, 2010), DARA (Abbasi *et al.*, 2007), PADRA (Akkaya *et al.*, 2010), DPCRA (Ranga *et al.*, 2014). In these approaches, there is a sudden increase in the MTR with the growth in network size. However, such an increase in the MTR is not present in our approach. The maximum numbers of messages transmitted are 24 for 100 nodes which is a very less number. Therefore our proposed algorithm recovers the network with less message overhead. From the figure, it can be determined that MTR increases when the network grows. However, MTR also remains the same in some cases, such as it is equal for (20, 30), (40, 50, 60), and (80, 90). Because this parameter also depends on the number of OHN of failed cut-vertex. E.g., if OHN of failed cut-vertex is the same for 80 nodes and 90 nodes, then the value for MTR will be equal in both the cases.

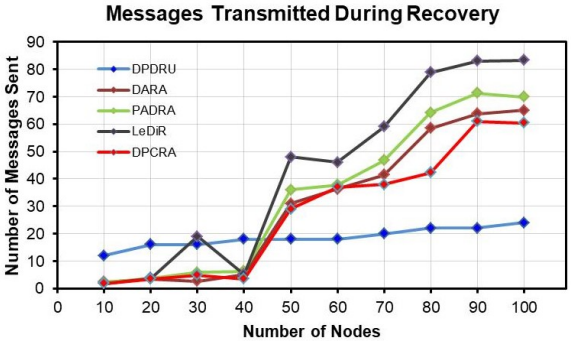

Fig. 11. MTR w.r.t number of nodes

3. **Distance Travelled by UAV for Recovery (DTUR):** Limited energy of UAV is an important constraint that limits deploying UAVs in various applications. More energy gets dissipated when UAV moves at a particular altitude. Therefore this DTUR parameter also signifies the energy dissipation of UAV during the recovery process. In this approach, distance traveled by UAV grows with more number of nodes, as shown in Figure 12. However, this parameter depends on the gap between the UAV and the failed node. Therefore, in some cases in Figure 12 distance traveled by UAV for 10 nodes and 90 nodes is less than the distance traveled by UAV for 20 nodes and 100 nodes, respectively. The velocity of UAV used for collecting these results is 10m/s.

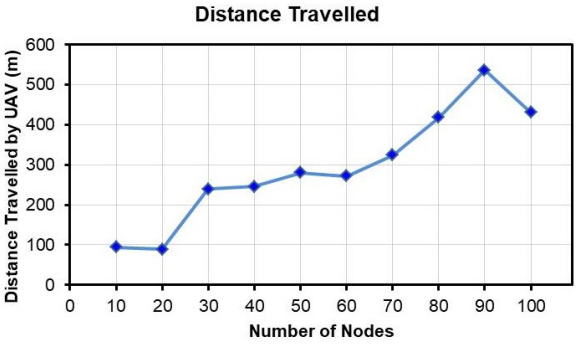

Fig. 12. DTUR w.r.t number of nodes

4. **Network Recovery Time (NRT):** NRT is an essential parameter for testing the performance, and it is equal to the time during which the network recovers. It is the time during which the sink node determines the failure of cut-vertex, commands UAV to reach the location of cut-vertex, and when it receives a recovery report from failure reporting nodes. All the periods involved in these steps are essential; however, the maximum time is utilized during the movement of UAV to the location of failed cut-vertex. Therefore NRT is directly proportional to the DTUR, as shown in Figure 13. This recovery time also varies with the velocity of UAV, with which it covers the whole distance after getting command from the sink. Thus Figure 14 shows the relationship between NRT and the velocity of UAV. NRT is indirectly proportional to the velocity of UAV. Figure 15 shows the NRT performance NRT concerning the network size. NRT grows with an increase in network size. However, it also depends on the arrangement of nodes, i.e., the distance between the failed cut-vertex and UAV, time taken by network nodes to send recovery reports. The NRT of 60 nodes and 100 nodes is less than the NRT of 50 nodes and 90 nodes. More NRT for fewer nodes can be related to Figure 12 where distance traveled by UAV is

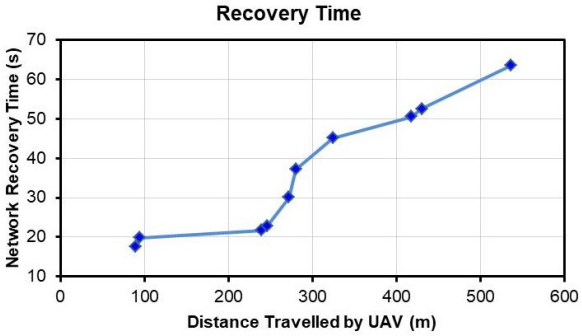

Fig. 13. NRT w.r.t distance traveled by UAV

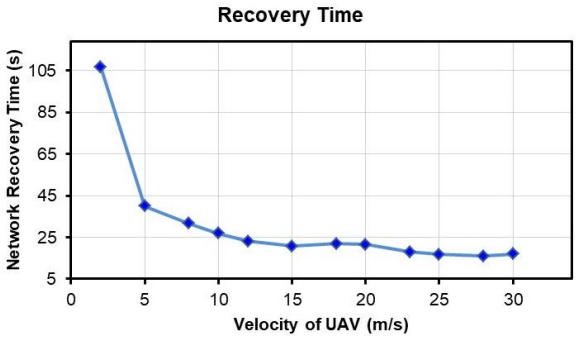

Fig. 14. NRT w.r.t Velocity of UAV

less for 60 nodes and 100 nodes as compared to 50 nodes and 90 nodes. The velocity of UAV used to collect the results of Fig 13 and Figure 15 is 10 m/s.

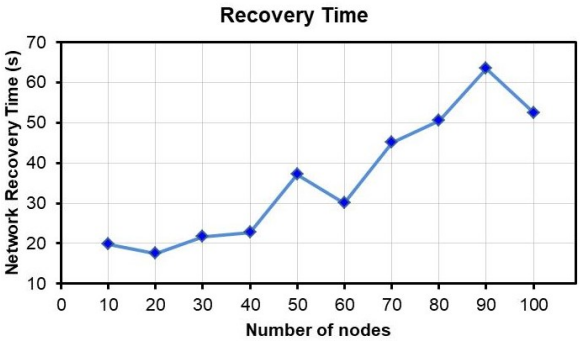

Fig. 15. NRT w.r.t number of nodes

7. Conclusion

This paper describes the proposed work on the detection and recovery of portioned WSANs. Our proposed approach Distributed Partition Detection and Recovery using UAV consist of three phases: Initialization, Operation and Detection phase, and Recovery phase. The algorithm focuses on reducing the computation and message overhead on the sink node by distributed detection of failure and sending sink node only important update messages (network information, failure report, and recovery report). The network recovery is performed with the help of UAV; thus, the energy of another network nodes remain conserved from additional movements of these nodes for network reconnection. The algorithmic analysis of the algorithm is done to analyze its efficacy. Experimental results are collected based on parameters such as Detection time, Network Recovery Time, distance traveled by UAV, and the number of messages transmitted. The message overhead is also compared with other similar approaches. In future work, we will try to improve the path traveled by UAV and improve the proposed algorithm to handle multiple node failures. We will also focus on the placement of UAVs in the 3-D environment in our research work, which is yet not seen in any literature. The security aspect is another area of exploration in the future which can halt the dispatch of the UAV for recovery.

ACKNOWLEDGMENTS

We are grateful to the Science and Engineering Research Board (SERB) - Department of Science and Technology (DST), Government of India, for sponsoring this research project.

References

Abbasi, A. A., Akkaya, K. & Younis, M. (2007), A Distributed Connectivity Restoration Algorithm in Wireless Sensor and Actor Networks, *in* '32nd IEEE Conference on Local Computer Networks (LCN 2007)', IEEE, pp. 496–503.

Abbasi, A., Younis, M. & Baroudi, U. (2010), 'Restoring connectivity in wireless sensor-actor networks with minimal topology changes', *IEEE International Conference on Communications* pp. 1–5.

Afzaal, H., Zafar, N. A. & Alhumaidan, F. (2017), 'Hybrid subnet-based node failure recovery formal procedure in wireless sensor and actor networks', *International Journal of Distributed Sensor Networks* **13**(4) 155014771770441.

Akkaya, K. & Senel, F. (2009), 'Detecting and connecting disjoint sub-networks in wireless sensor and actor networks', *Ad Hoc Networks* **7**(7) 1330–1346.

Akkaya, K., Senel, F., Thimmapuram, A., Uludag, S., Hwang, S., Chao, W., Wu, C. & Dow, C. (2010), 'Distributed Recovery from Network Partitioning in Movable Sensor/Actor Networks via Controlled Mobility', *IEEE Transactions on Computers* **59**(2) 258–271.

Alfadhly, A., Baroudi, U. & Younis, M. (2011), 'Least Distance Movement Recovery approach for large scale wireless sensor and actor networks', *IWCMC 2011 - 7th International Wireless Communications and Mobile Computing Conference* pp. 2058–2063.

Bayrakdar, M. E. (2020a), 'Cooperative communication based access technique for sensor networks', *International Journal of Electronics* **107**(2) 212–225.

Bayrakdar, M. E. (2020b), 'Employing sensor network based opportunistic spectrum utilization for agricultural monitoring', *Sustainable Computing: Informatics and Systems* **27**, 100404.

Bayrakdar, M. E. (2020c), 'Energy-efficient technique for monitoring of agricultural areas with terrestrial wireless sensor networks', *Journal of Circuits, Systems and Computers* **29**(09) 2050141.

Bayrakdar, M. E. (2020d), 'Enhancing sensor network sustainability with fuzzy logic based node placement approach for agricultural monitoring', *Computers and Electronics in Agriculture* **174**, 105461.

Bayrakdar, M. E. (2020e), 'Exploiting cognitive wireless nodes for priority-based data communication in terrestrial sensor networks', *ETRI Journal* **42**(1) 36–45.

Cheng, M. X., Ling, Y. & Sadler, B. M. (2017), 'Network connectivity assessment and improvement through relay node deployment', *Theoretical Computer Science* **660**, 86–101.

Chouikhi, S., Korbi, I. E., Ghamri-Doudane, Y. & Azouz Saidane, L. (2017), 'Centralized connectivity restoration in multichannel wireless sensor networks', *Journal of Network and Computer Applications* **83**(November 2016), 111–123.

Chriki, A., Touati, H., Snoussi, H. & Kamoun, F. (2019), 'FANET: Communication, mobility models and security issues', *Computer Networks* **163**, 106877.

Devi, E. A. & Manickam, J. M. L. (2014), Detecting and repairing network partition in wireless sensor networks, *in* '2014 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2014]', IEEE, pp. 1338–1343.

Han, X., Cao, X., Lloyd, E. L. & Shen, C.-C. (2009), 'Fault-tolerant relay node placement in heterogeneous wireless sensor networks', *IEEE Transactions on Mobile computing* **9**(5) 643–656.

Hashim, H. A., Ayinde, B. O. & Abido, M. A. (2016), 'Optimal placement of relay nodes in wireless sensor network using artificial bee colony algorithm', *Journal of Network and Computer Applications* **64** 239–248.

Hwang, S., Chao, W., Wu, C. & Dow, C. (2014), 2-Connected relay node placement scheme in disjoint wireless sensor networks, *in* '2014 IEEE 5th International Conference on Software Engineering and Service Science', pp. 1039–1043.

Imran, M., Younis, M., Said, A. M. & Hasbullah, H. (2010), Partitioning detection and connectivity restoration algorithm for wireless sensor and actor networks, *in* '2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing', IEEE, pp. 200–207.

Jha, V., Prakash, N. & Mohapatra, A. K. (2019), 'Energy Efficient Model for Recovery from Multiple Nodes Failure in Wireless Sensor Networks', *Wireless Personal Communications* **108**(3) 1459–1479.

Joshi, Y. K. & Younis, M. (2014), Mobility-based internetworking of disjoint segments, *in* '2014 27th Biennial Symposium on Communications (QBSC)', IEEE, pp. 193–197.

Joshi, Y. K. & Younis, M. (2016), 'Exploiting skeletonization to restore connectivity in a wireless sensor network', *Computer communications* **75**, 97–107.

Kumar, R. & Amgoth, T. (2019), 'Adaptive cluster-based relay-node placement for disjoint wireless sensor networks', *Wireless Networks* **8**.

Lalouani, W., Younis, M. & Badache, N. (2017), 'Optimized repair of a partitioned network topology', *Computer Networks* **128**, 63–77.

Lee, S. & Younis, M. (2010a), 'Optimized relay placement to federate segments in wireless sensor networks', *IEEE Journal on Selected Areas in Communications* **28**(5) 742–752.

Lee, S. & Younis, M. (2010b), 'Recovery from multiple simultaneous failures in wireless sensor networks using minimum Steiner tree', *Journal of Parallel and Distributed Computing* **70**(5) 525–536.

Lee, S., Younis, M. & Lee, M. (2015), 'Connectivity restoration in a partitioned wireless sensor network with assured fault tolerance', *Ad Hoc Networks* **24**(PA), 1–19.

Lee, S., Younis, M. & Lee, M. (2016), 'Optimized bi-connected federation of multiple sensor network segments', *Ad Hoc Networks* **38**, 1–18.

Liu, S., Steinert, R. & Kostic, D. (2018), Control under Intermittent Network Partitions, *in* '2018 IEEE International Conference on Communications (ICC)', Vol. 2018-May, IEEE, pp. 1–7.

Ma, G., Yang, Y., Qiu, X. & Gao, Z. (2016), 'Obstacle Aware Connectivity Restoration for Disjoint Wireless Sensor Networks Using a Mix of Stationary and Mobile Nodes', *International Journal of Distributed Sensor Networks* **12**(5) 6469341.

Mahmood, K., Khan, M. A., Hassan, M. U., Shah, A. M., Ali, S. & Saeed, M. K. (2018), 'Intelligent On-Demand Connectivity Restoration for Wireless Sensor Networks', *Wireless Communications and Mobile Computing* **2018**, 1–10.

Nikolov, M. & Haas, Z. J. (2016), 'Relay placement in wireless networks: Minimizing communication cost', *IEEE Transactions on Wireless Communications* **15**(5) 3587–3602.

Ranga, V., Dave, M. & Verma, A. K. (2013), 'Network partitioning recovery mechanisms in WSANs: A survey', *Wireless Personal Communications* **72**(2) 857–917.

Ranga, V., Dave, M. & Verma, A. K. (2014), 'A hybrid timer based single node failure recovery approach for wsans', *Wireless personal communications* **77**(3) 2155–2182.

Ranga, V., Dave, M. & Verma, A. K. (2016a), 'Optimal nodes selection in wireless sensor and actor networks based on prioritized mutual exclusion approach', *Kuwait Journal of Science* **43**(1) 150–173.

Ranga, V., Dave, M. & Verma, A. K. (2016b), 'Restoration of lost connectivity of partitioned wireless sensor networks', *Cogent Engineering* **3**(1) 1–22.

Senel, F. & Younis, M. (2011a), Optimized connectivity restoration in a partitioned wireless sensor network, *in* '2011 IEEE global telecommunications conference-GLOBECOM 2011', pp. 1–5.

Senel, F. & Younis, M. (2011b), 'Relay node placement in structurally damaged wireless sensor networks via triangular steiner tree approximation', *Computer Communications* **34**(16) 1932–1941.

Senturk, I. F., Akkaya, K. & Yilmaz, S. (2014), 'Relay placement for restoring connectivity in partitioned wireless sensor networks under limited information', *Ad Hoc Networks* **13**(PART B), 487–503.

Senturk, I. F., Yilmaz, S. & Akkaya, K. (2012), 'A game-theoretic approach to connectivity restoration in wireless sensor and actor networks', *IEEE International Conference on Communications* pp. 7110–7114.

Shriwastav, S. & Ghose, D. (2018), 'Round-table negotiation for fast restoration of connectivity in partitioned wireless sensor networks', *Ad Hoc Networks* **77**, 11–27.

Srinivasaperumal, M., Raja, K. B., Balaji, G. N. & Dally, E. C. (2017), 'Concurrent Node Recovery From Failure In Wireless Sensor-Actor Networks', (February).

Stojmenovic, I., Simplot-Ryl, D. & Nayak, A. (2011), 'Toward scalable cut vertex and link detection with applications in wireless ad hoc networks', *IEEE Network* **25**(1) 44–48.

Verma, G. K. & Ranga, V. (2018), 'Whale optimizer to repair partitioned heterogeneous wireless sensor networks', *International Journal of Grid and Distributed Computing* **11**(5) 11–28.

Younis, M., Senturk, I. F., Akkaya, K., Lee, S. & Senel, F. (2014), 'Topology management techniques for tolerating node failures in wireless sensor networks: A survey', *Computer Networks* **58**(1) 254–283.

Zahid, S., Abid, S. A., Shah, N., Abbas Naqvi, S. H. & Mehmood, W. (2018), 'Distributed Partition Detection With Dynamic Replication Management in a DHT-Based MANET', *IEEE Access* **6**, 18731–18746.

Zhang, J. & Liu, K. (2019), 'Survey of Ad-Hoc network technology for UAV', *International Conference on Communication Technology Proceedings, ICCT* **2019-Oct**. (1) 260–265.

Zhang, Y., Wang, J. & Hao, G. (2018), 'An Autonomous Connectivity Restoration Algorithm Based on Finite State Machine for Wireless Sensor-Actor Networks', *Sensors* **18**(2) 153.

Submitted: 24/10/2020
Revised: 16/12/2020
Accepted: 24/12/2020
DOI: 10.48129/kjs.v48i4.10819