Wireless Pers Commun (2013) 72:857–917
DOI 10.1007/s11277-013-1046-7

Network Partitioning Recovery Mechanisms in WSANs: a Survey

Virender Ranga · Mayank Dave · Anil Kumar Verma

Published online: 21 February 2013
© Springer Science+Business Media New York 2013

Abstract Wireless sensor and actor networks (WSANs) are more promising and most addressing research field in the area of wireless sensor networks in recent scenario. It composed of possibly a large number of tiny, autonomous sensor devices and resources rich actor nodes equipped with wireless communication and computation capabilities. Actors collect sensors' information and respond collaboratively to achieve an application specific mission. Since actors have to coordinate their operation, a strongly connected inter-actor network would be required at all the time in the network. Actor nodes may fail for many reasons (i.e. due of battery exhaustion or hardware failure due to hash environment etc.) and failures may convert connected network into disjoint networks. This can hinder sometimes not only the performance of network but also degrade the usefulness and effectiveness of the network. Thus, having a partitioning detection and connectivity restoration procedure at the time of failure occurs in the network is crucial for WSANs. In this paper, we review the present network partitioning recovery approaches and provide an overall view of this study by summarizing previous achievements.

Keywords WSANs · Optimization criteria · DARA · LeMoToR · LeDir · PADRA · BiO4SeL

V. Ranga (✉) · M. Dave
Department of Computer Engineering, National Institute of Technology, Kurukshetra, Haryana, India
e-mail: virendersinghmtech@gmail.com; virender.ranga@nitkkr.ac.in

M. Dave
e-mail: mdave67@gmail.com

A. K. Verma
Department of Computer Science and Engineering, Thapar University, Patiala, Punjab, India
e-mail: akverma@thapar.edu

1 Introduction

In recent years, due to development in wireless sensor technologies, wireless sensor networks (WSNs) have witnessed a paradigm shift from traditional homogeneous to heterogeneous deployments containing a mixing of resource constrained sensor nodes and powerful nodes known as actors. The capabilities of actor nodes include enhanced processing ability, powerful wireless transmission, large battery back-up and mobility capability. Actor nodes can leverage their capabilities to facilitate better coverage, connectivity and energy efficient operation. For instance, actors with controlled mobility can be used to mend a partitioned network, set up new routes via connectivity, facilitate energy efficient data collection and improve sensor coverage. However, there are some complex scenarios that require cooperation between sensors and higher capability devices, such as actor nodes to support the proper execution of specific tasks [1,2].

Nowadays, wireless sensor and actor networks (WSANs) are widely used in various areas of human life, especially those serving in remote and harsh environments in which human intervention is risky or impractical. One specific use case of these networks is providing communication services between the members of rescue team in emergence search and rescue such as disaster scenarios. In such cases, saving of human life may be depending on the proper functioning of the communication network so that information about the survivors can be delivered correctly, and accordingly rescue operation may be launched [3].

A typical problem of WSANs is network partitioning problem due to failure of one or more actor nodes i.e. when some part of the network isolates from remaining network due to failure of critical node(s). This problem is more general problem due to random and sparse deployment of actor nodes in the network. This problem (i.e. cut-vertex nodes' failure) is different from link's failure/path failure in the sense that network is isolated in to disjoint parts due to critical nodes are physically failed [i.e. due to battery exhaustion or physically damaged due to some natural phenomena (i.e. sand storm, volcano, earth quakes or tsunami etc.)], whereas link's failure(s) means communication error due to some node interference or channel congestion or nodes' buffer overflow etc. for time being and recovery occurs after choosing alternate paths or applying congestion control mechanism/error control mechanism. Routing protocols deal with link's failure, if some alternate routes are available in the network (i.e. it is assumed in the network). Therefore, network partitioning problem (i.e. failure of cut-vertex node(s)) is most challenging research area in WSANs.

To recover from the network partitioning problem, sometimes additional mobile nodes can be used such as unmanned Aerial Vehicles (UAVs) or robots. But these mobile nodes are very expensive and sometimes difficult to deploy due to terrain constraint, and take more time to deploy. For cost effectiveness, instead of taking some additional mobile nodes explicitly, the network should be self-healing (i.e. without any human intervention, the network assist autonomously itself about the failure of critical nodes and recovery takes place using controlled mobility of actor nodes) so that implicit nodes can be used for network partitioning recovery. In addition, recovery of failed node should be energy efficient, lightweight in terms of required communication and computational resources, and within specified time.

The main objective of this work is to provide a common framework for the art of study of this research area (i.e. network partitioning problem) and compared proposed approaches with different performance parameters used in recovery process. The remainder of this paper is organized as follows. Section 2 presents a brief description of network architectures and models. Section 3 outlines the design issues. Section 4 provides brief explanation of current and future applications. Section 5 explains future research challenges. Section 6 provides a novel taxonomy of state-of-the-art approaches based upon their approach type, level of

failure(s) in the network (i.e. single node's failure or multi-node failures). Section 7 provides a comparative study to understand various performance metrics to improve robustness in present scenario. Finally, Sect. 8 concludes and gives future direction of our work.

2 WSAN Architectures and Models

Two different types of architectures can be defined according to the way the data is collected by the sensor nodes and reports back to the actor nodes i.e. automated and semi-automated [4,5]. In automated architecture, data is collected by the sensor nodes and transmit directly to the actor nodes, which will efficiently coordinate to execute a specific task without collaboration from the sink. As a result, automated architecture is recommended for time sensitive applications where a fast reaction by the actors is a critical requirement.

In semi-automated architecture, the sensor data transmit to a central controller (e.g. the sink) which will process the collected data and will determine which actors take action to execute a specific task; this is accomplished by transmitting a set of commands to the corresponding actors.

Furthermore, WSANs have two communication models i.e. single-actor and multi-actor models. In the single-actor model, sensor readings may be sent only to one actor node for execution of task. In this way, if the actor has enough capability to perform the action, it can immediately do so, especially for single-actor task. This implies that in this model latency between sensing and acting may be low as well as there is no needed of actor-actor coordination [3]. Even if one actor is not sufficient for the required action, the same can publish the announcement message to other actors. After it receives responses from other actors, it selects one or some of the available actors and let it/them to perform the action. Therefore, this model may cause high network load as well as high latency.

Moreover, WSAN architectures require implementation of multiple coordination levels i.e. **Sensor-Sensor (SS)**, **Sensor-Actor (SA)** and **Actor-Actor (AA)**. The SS coordination is employed to gather information from the physical world in an effective and energy efficient way [6,7]. On the other hand SA is required to communicate the intercepted information reliably to actors. In addition, SA coordination may also be used over the downlink (i.e. from the actor toward the sensor) to inform the sensors to proceed with specific sensing tasks. The Actor-to-Actor coordination is required to execute a specific task while coordinating which actor nodes that respond within a certain area.

3 Design Issues in WSANs

In this section, we broadly classify the design issues under the following categories [8,9].

3.1 Network Dynamics

Routing messages from or to moving nodes is more challenging since route stability or node stability becomes an important optimization factor, in addition to energy, bandwidth etc. The sensed event can be either dynamic or static depending on the application. For instance, in a target detection/tracking application, the event (phenomenon) is dynamic whereas forest monitoring for early fire prevention is an example of static events. Monitoring static events allow the network to work in a reactive mode, simply generating traffic when reporting. Dynamic events in most applications require periodic reporting and consequently apply

action according to the sensed data. Therefore, to handle network dynamics in WSANs is a challenging issue.

3.2 Node Deployment

Another issue is topological deployment of actor nodes with sensor nodes. This is an application dependent and affects the performance of the routing protocol. The deployment is either deterministic or self-organizing. In deterministic situations, sensors and actors are manually placed and data is routed through predetermined paths. However, in self-organizing systems like WSANs, the sensor nodes are scattered randomly creating an infrastructure in an ad hoc manner. In that infrastructure, the position of the sink or the cluster-head is also crucial in terms of energy efficiency and performance. When the distribution of nodes is not uniform, optimal clustering becomes a pressing issue to enable energy efficient network operation.

3.3 Data Fusion

Sensor nodes generate significant redundant data in WSANs and similar packets from multiple nodes can be aggregated so that the number of transmissions would be reduced. Data fusion is the combination of data from different sources by using functions such as suppression (eliminating duplicates), min, max and average. Some of these functions can be performed either partially or fully in each sensor node, by allowing sensor nodes to conduct in-network data reduction. Recognizing that computation would be less energy consuming than communication, substantial energy savings can be obtained through data aggregation. Therefore, in WSANs, how to achieve energy efficient data fusion technique is a big issue due to heterogeneity nature of the network.

3.4 Network Lifetime

Another issue is to prolong the lifetime of network. WSANs are generally limited by battery lifetime of actor nodes and maximize network lifetime of network is certainly one of the most important design objectives for WSANs. Lifetime of a network can be defined as duration time which percentage of dead nodes below a threshold. The ability of a network to prolong network lifetime is typically evaluated in various literatures based on its definition [10–14].

3.5 Tolerating Bounded Packet Delay

Actors have to act on sensed data quickly. Otherwise, it would be detrimental to the operation of a WSAN. Specifically, any communication protocol as well as recovery protocol must ensure that packets do not exceed an application's delay tolerance. This means sensor nodes need to form low delay and reliable paths to one or more backbone nodes, and actor nodes must collaboratively to reach a consensus on the best response to one or more events.

3.6 Reliability

Reliability means to have correct execution of actions by actor nodes. In WSANs, actor nodes need to receive sensed data within a pre-determined time period in order to re-construct an event, understand its intensity, location and coverage, and lastly determine the appropriate

number of actors that are used in response to the event. Unfortunately, sensed data and commands may be lost due to congestion, bit error, or bad connectivity due to failure of critical nodes. Therefore, protocols must be developed to address one or more of these issues such that sensed data and control information are communicated to sensor and actor nodes fairly (without error) and reliably.

3.7 Mobility

The mobility nature in WSAN is completely different to WSNs. Mobile elements in WSNs are designed to save the energy of sensor nodes by collecting data from sensor nodes directly or via rendezvous or cache points. On the other hand, in WSANs, mobile nodes are used to reduce end-to-end packet delay and guarantee task completion times, distributed failure recovery etc.

4 Current and Future Applications of WSANs

A variety of current possible applications of WSANs are available from environmental monitoring [15], health care [16], positioning and tracking [17], to logistic, localization, and so on. A brief introduction of future applications is given in this section. It is important to underline that the application strongly affects the choice of the wireless technology to be used. Once application requirements are set, in fact, the designer has to select the technology which satisfies these requirements. To this aim, the knowledge of the features, advantages and disadvantages of the different technologies is fundamental requirement. Owing to the importance of the relationship between application requirements and technologies, we report some future applications of WSANs [18,19].

4.1 Global Scale Environmental Monitoring and Emergency Operations

Today, many sensors exist around the world collecting environmental data. In most cases, the systems focus on single problem. Most of these systems measure a limited number of parameters at a large granularity (sensors separated by 100 or 1,000 of meters). WSANs have the potential of dense and flexible coverage. When unexpected environmental events occur or when natural disasters destroy the current infrastructure (earthquakes, cyclones), this technology can be rapidly deployed for almost immediate collection of data. The major factors that favor this technology for such tasks are self-configuration of the system with minimal overhead and independent of fixed or centralized infrastructure.

4.2 Social Participatory Computing

The ubiquity of this technology that it includes devices that are carried by individuals as well as many emplaced systems in our surroundings. The access of real-time data will transform how people work, socialize, and go about their daily routines. Individuals will be able to track commuting delays and minimize or avoid problems associated with these delays. Municipalities will be able to control traffic patterns to reduce congestion. People will also be able to link to groups of individuals with similar interests and interact with them from anywhere at any time. Overall, individuals will become enmeshed with and rely on these WSANs for efficient and happier lifestyles.

4.3 Medical Care

One use of the widespread availability of WSANs will be for medical care. It will be possible to create an account for each person when they are born and via physiological, activity, and environmental sensing keep a record of their health and activities that relate to health. Using such information, preliminary diagnosis can be achieved without a doctor's visit. This will also enable preventative care. As one ages or as one's health deteriorates, specific devices can be included in smart clothes or within the home to ameliorate the condition and/or treat it more effectively. In some cases, automatic medical treatment might be administered.

4.4 Precision Agriculture

One use of the widespread availability of WSANs will be for precision agriculture, in which hundreds of nodes scattered throughout a field, establish a routing topology, and transmit data back to a collection point. The main function of the sensors is to collect the information about various factors like the moisture required for crops, which part of the field requires more fertilizer or more water etc and take action according to the requirement without any human intervention. This can also dictate control of pesticide and fertilizer amounts. If one of the nodes fails, a new topology would be established and the overall network would continue to deliver data. Therefore, this type of application demands more robustness, scalability, low-cost and easy to deploy networks.

5 Challenges and Open Research Issues in WSANs

In order to develop a platform which provides network partitioning problem free system, many challenges and open research issues need be faced and solved. Many of them are common to WSNs but there are some that arise from the intrinsic nature that these devices currently present.

5.1 Resource Limited Platform

The capacity of current batteries used to power sensor nodes is expected to grow as time goes by, but not fast enough to satisfy the sensor node demands. All protocols developed for these kinds of platforms need to be designed to be energy efficient. Because the most energy expensive operation involves communicating sensor nodes using the radio transceiver special emphasis should be put on minimizing data communication. If a node goes down as a result of a depleted battery, it is not always feasible to manually replace it. Alternatively, some node partitioning recovery techniques can be used in order to ensure connectivity. Monitoring applications in WSANs usually relay all the information they get from the sensors to some sink nodes. As a result, memory consumption in the nodes of the networks will be unbalanced, which means, nodes near the sink nodes will receive more packet traffic than the rest. Some techniques need to be developed to balance the network traffics to the best extent possible.

5.2 Cross Layer Architecture

The current WSN and WSAN protocols design are largely based on layered approach. The layered design makes it possible to change or update some parts of it without affecting the whole protocol. However, the inflexibility and sub optimality cause poor performance of

WSANs due to constraints of low energy consumption and low latency. Therefore, instead of having individual layers, cross layer design allows to communicate layers (such as physical and network layers) which traditionally have been independent in order to exchange more information that can be used to improve efficiency. However, leveraging a cross layer approach can provide much more effective sensing, data transmission, and acting in WSANs. Several cross-layer integration issues among the communication layers should be investigated in order to improve the overall efficiency of WSANs.

5.3 n-to-N Communication

Communications in WSANs are not restricted to sensor reading delivery between sensors and sink nodes. More complex communications are also used in many proposals which involve an n-to-N communication pattern between all devices in the network. This is particularly true in networks which not only report information to the sink nodes but also can be managed and accessed by users. Although, n-to-N communications improve the system flexibility and capabilities, but this kind of communication make it more difficult to guarantee QoS. Designing such protocols that provide QoS requirements is very challenging.

5.4 Topology Management

WSANs are highly dynamic distributed systems. The topology of the network is expected to change as nodes fail, or added or their internal state changes (from or to sleep mode for example). Also, the connectivity graph can change as the communication range varies because of a change in the transmission power or due to external factors. In addition, if node mobility is allowed, the protocols are much more complex. Designing protocols that allow topology organization/re-organization is very challenging. The fact that the position of a node can vary over time makes it hard to route packets under strict QoS requirements. Routing tables are much more dynamic and neighbor discovery techniques become much more complex especially if the node mobility range is not bounded.

6 Background Study

To the best of our knowledge, there is no specific taxonomy for network partitioning recovery mechanisms for WSANs till date. Although, there is some related work like; Akyildiz et al. [3] which presents WSANs challenges for future direction and Younis et al. [20] presents strategies and techniques for node placement in WSNs. However, these works considered only actor to actor coordination (AOA), actor to sensor coordination (AOS), node placement techniques etc.; they do not include any nodes' failure recovery mechanism for network partitioning problem. Unlike, I.F. Akyildiz et al. and M. Younis et al., Fig. 1 shows the novel taxonomy of network partitioning recovery approaches. In the first section, we have studied the comprehensive study of single node's failure recovery approaches and in second section multiple nodes' failure recovery approaches are explained based on existing state-of-the-art literature review. Furthermore, we have classified all the approaches based upon their nature (i.e. distributed or centralized), their behavior in the network (i.e. proactive, reactive or hybrid) and level of failure(s) occurs in the network i.e. single node's failure or two node failures (special case of multi-node failures) or large scale failures after summarizing the previous achievements.

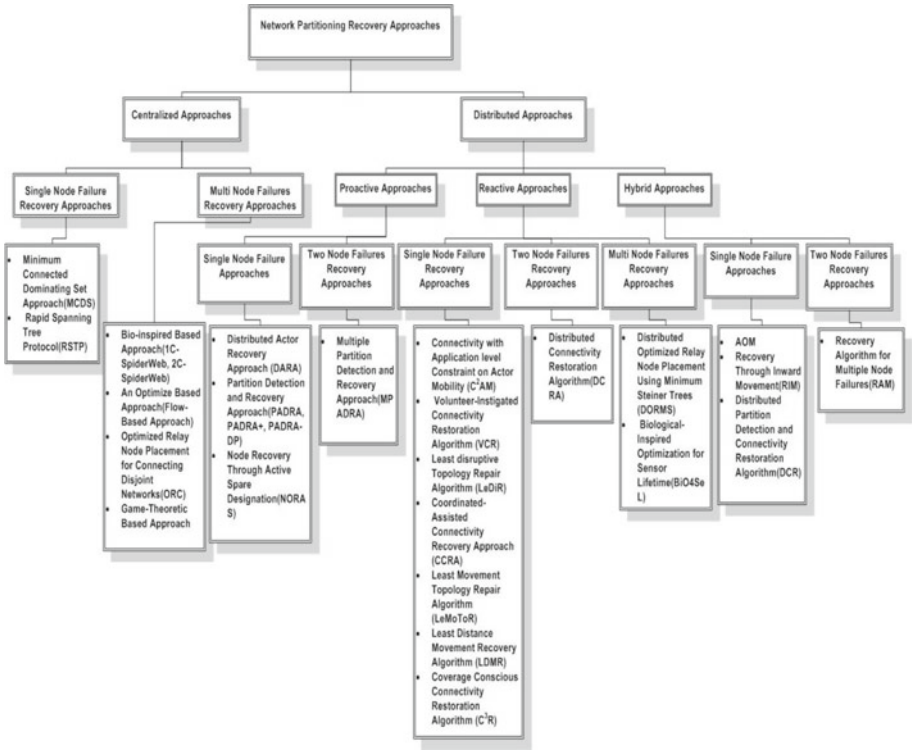

Fig. 1 Taxonomy of network partitioning recovery approaches

6.1 Single Node Failure Recovery Approaches

This section gives brief introduction of single nodes' failure approaches based on state-of-the-art literature surveyed. In such type of failure, it is considered that one node is failed at a time and recovery can be self-healing (i.e. autonomously repair) and energy efficient to prolong network lifetime.

6.1.1 Minimum Connected Dominating Set Approach (MCDS)

Senel et al. [21] described first centralized algorithm for establishing connected inter-actor network with intra connected sub-networks. The idea is to pursue a coordinated actor movement in order to connect the sub-networks with minimize total travelled distance and maximum travel distances of the individual actors. The authors considered the minimum connected dominating set of each sub-network while picking the appropriate actor to move so that the connectivity of each sub-network is not violated. However, such movement may disconnect the repositioned actors from their sub- networks; therefore, authors considered the additional intermediate nodes for performing cascaded movements in order to maintain the intra sub-network connectivity. Moreover, moving a node for a relatively long distance can drain significant amount of energy, therefore a cascaded movement is preferred, if there are sufficient number of sensors on the way. The objective of the cascaded movements is

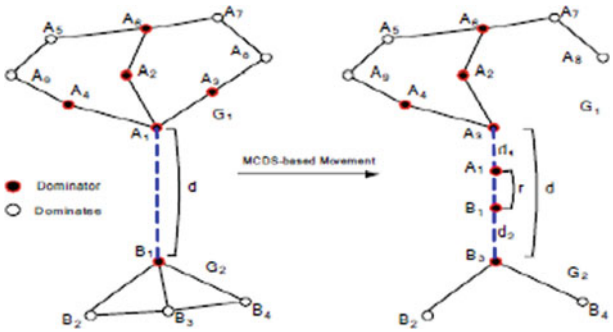

Fig. 2 Connecting two sub-networks G1 and G2 after the failure of a critical node [21]

threefold: (a) To minimize the number of movements and thus total travelled distance of the involved actors. (b) To minimize the maximum distance an actor will travel. (c) To distribute the movement load among the participated actor nodes such that the energy of involved actors may not get quickly depleted, thus extending the network life. Two steps are given for the recovery of network partitioning:

(i) *Detecting the partitions:* It is assumed that nodes are deployed uniformly and there is no coverage hole in the network. After the initial deployment, each sub-network looks for the other sub-networks in the area of interest which are not connected to it. Each node can send heartbeat message to its neighbor's node to check whether node is alive or not. If the node would not get the acknowledgement form the neighbor node, then it is assumed that node gets failed and there is network partition in the network.
(ii) *Establishing connectivity:* After the partitions detection in the network, any two sub-networks can be connected by moving the closest nodes from each sub-network towards each other and filling in the space for restoration of connectivity.

To understand the proposed solution, let us consider two sub-networks G1 and G2 which are disconnected after the node failure as shown in Fig. 2. First solution for recovery may be to reconnect both the sub-networks, G1 and G2 needs to reposition along the line such that nodes remain connected to either G1 or G2 and have a distance less than r. In such case, the minimum distance between two nodes will need to move $\frac{(d-r)}{2}$.This is the minimum distance travelled by individual node for both G1 and G2 as shown in Fig. 2.

Another solution for recovery is to pick a number of actor nodes from one or both of the sub-networks whose absent does not violate the connectivity in both G1 and G2 and relocate along the line of closest actor nodes. However, the moving distance of an actor in this case can be very large and moving distance of individual actor will be unacceptably higher than $\left(\frac{(d-r)}{2}\right)$.

Both solutions are not acceptable due to large movements of actor nodes which cause the moving nodes to die rather quickly as compared with other actor nodes in the network. Therefore, the authors have proposed a hybrid solution which combines the advantages of both the solutions. The authors have taken two optimized rules:

- Minimizing total moving distance of involved actor nodes for recovery through cascaded relocations whenever is necessary (i.e. TMI).
- Maintain a maximum of $\left(\frac{(d-r)}{2}\right)$ units of movements for individual actor nodes (i.e. MMI).

The cost function is given below:

$$Min \sum_{\forall i \in A} M_i$$

Subject to

$$\forall i \in A$$
$$M_i < (d - r)/2$$

where $A \in (A_1, A_2 \ldots\ldots A_n)$. M_i movement of ith node, r communication range of an actor node.

The main problem in MCDS approach is to choose a candidate to replace failed node for recovery. This approach does not take into account whether the selected actor is critical or not. The blind movement may cause repartitioning of network again, which is inefficient and also increase the restoration time as well as cause lot of movements of nodes i.e. criticality is not considered in this approach.

6.1.2 Distributed Actor Recovery Algorithm (DARA)

Abbasi et al. [22] suggested a localized, distributed real-time restoration algorithm to restore the connectivity of inter-actor network that has been affected by the failure of an actor node and avoids the involvement of every single actor node in the network. The idea is to replace the dead actor node by a suitable candidate based on nodes' degree (i.e. how many actor nodes are in the 1-hop neighbors) and physical proximity to the dead node. The optimized objective is to minimize the total distance travelled by the involved actors in order to limit the overhead incurred by the movements. The authors have considered three criteria in the order given below:

- Minimum Node Degree (MND).
- Minimum distance to failed actor node (MDA).
- To break the tie use highest Node_Id before lowest Node_Id node.

To understand this approach, consider a network topology as shown in Fig. 3a. In this, A_1 is considered as cut-vertex; because the failure of this node converts the network into multiple disjoint sub-networks as shown in Fig. 3b. The failure of A_1 will cause the partitioning of the network into 3 disjoint sub-networks, namely $\{A_2, A_3\}$, $\{A_4, A_5, A_6, A_7, A_8\}$ and $\{A_9, A_{10}, A_{11}, A_{12}, A_{13}, A_{14}, A_{15}\}$. Instead of running a block movement, DARA pursues a cascaded relocation of few actors.

We argue that cascaded relocation requires less number of movements when compared to block movement which rather causes all the actors in the sub-network to move. In addition, block movement requires all the actors in the sub-network to be aware of where and how far to move which introduces extra messaging. DARA only requires that each actor forms a 2-hop neighbor lists which can be easily maintained.

DARA expects nodes A_2, A_7, A_8 and A_9 to initiate the recovery procedure since they are immediate neighbors to A_1. The basic idea of DARA is that one of the immediate neighbors of the faulty actor will relocate to where A_1 is to reconnect the disjoint partitions. Clearly three issues exist: (a) how we prevent further partitioning conditions that result from moving nodes, (b) which of the neighbors will move, (c) how we ensure that only one actor will replace the failed actor.

To deal with the first issue, DARA pursues cascaded node relocation in order to sustain connectivity. DARA establishes selection criteria that are based on the node degree and the

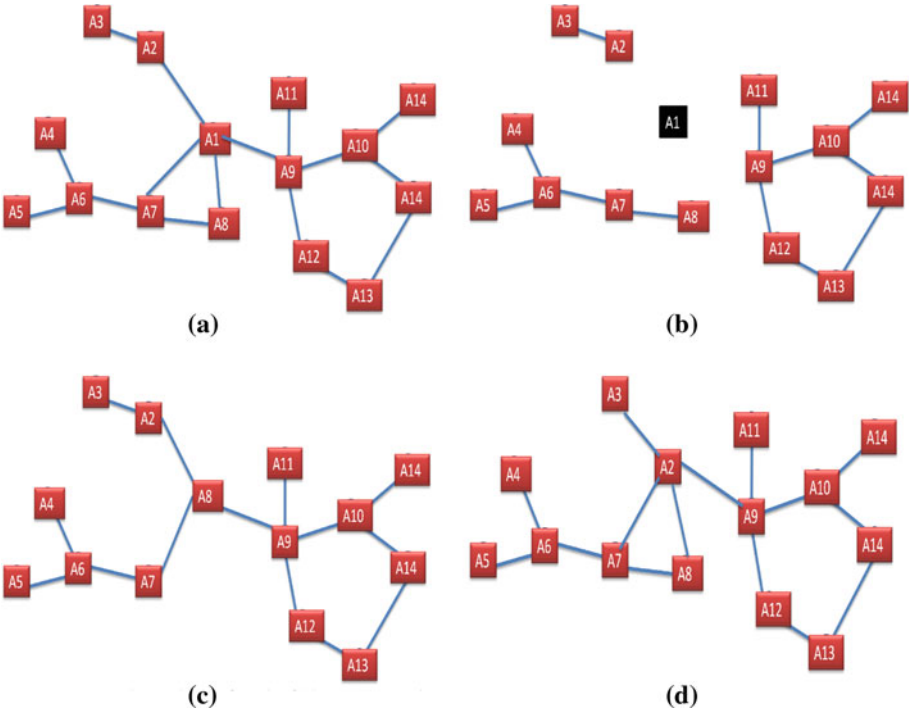

Fig. 3 **a** Connected inter-actor network. **b** Three disjoint sub-networks after the failure of A_1. **c** Network topology after the failure recovery by A_8. **d** Network topology after the failure recovery by A_2 and A_3

proximity of the neighbor. DARA prefers the node(s) that has fewer neighbors (children) since the scope of the cascaded relocation will be limited. Legible choices are filtered based on their proximity to the failed node since the travel distance will be shorter and thus the overhead will be limited. In case of ties, the Node_Id is used for final arbitration. Since every actor is aware of its 2-hop neighbors, nodes A_2, A_7, A_8 and A_9 would independently come to the same conclusion and only one of them will assume responsibility for conducting the recovery. Among A_2, A_7, A_8 and A_9, A_9 has the highest node degree and will be thus excluded. Also, A_7 has node degree of two which is larger than that of A_2 and A_8. Since A_2 and A_8 have the same node degree and are equidistant to A_1, node A_8 is picked based on the Node_Id. Figure 3c shows the network topology after the recovery. It is worth noting that if A_2 is to be picked, A_3 may also need to move as shown in Fig. 3d and thus the total travel distance of all involved actors will be longer than the case when node A_8 is selected. Therefore, DARA may not always yield the optimal results, which is typical for a greedy approach.

This approach is good for single node failure recovery, but it has following short comes:

- DARA does not provide any mechanism to detect the cut-vertices. It assumes this information is available at the node which may require the knowledge of whole topology.
- The selection of failure handler (FH) to replace the failed node is done based on the neighbor's degree and distance to failed node which may requires excessive replacement until a leaf node is found. Sometimes, moving distance may be large.

6.1.3 Partition Detection and Recovery Algorithm (PADRA, PADRA+ and PADRA-DP)

Akkaya et al. [23,24] proposed a distributed proactive approach by pre-planning the failure handling process. The rationale of approach is that the neighbors of a cut-vertex will not be able to communicate after the failure of that node and some node(s) is required for handling such failure. PADRA preferred the closest node that has a dominatee. If no such dominatee is available, then closest node i.e. MMI will be preferred as its failure handler (FH) and apply same procedure on that node until any dominatee is hit. If the neighbor/s of the failed node is/are all dominator/s, then failed node prefers the closest one that has a dominatee among its neighbors. The authors have considered two optimization rules for recovery process:

- Minimum travel distance of individual node i.e. MMI.

$$Min\,\{M_i\}$$

- Minimum total movement distance of all actor nodes i.e. TMI.

$$Min \sum_{\forall i \in S} M_i$$

Subject to

$$\forall i \in S$$
$$M_i < r_i$$

where $S \in \{S_1, S_2 \ldots\ldots S_n\}$, M_i movement of ith node, r_i communication range of actor node.

Moreover, the authors have also taken the concept of dynamic programming (DP) to find the closest neighbor dominatee. The idea is to find the least cost path to the closest dominatee. Therefore, the failure handler (FH) nodes start a search process among the sub-tree of its neighbors in order to find the closest dominatee. Basically each sub-tree returns its cost of reaching a dominatee to the failure handler (FH). The failure handler picks the least cost dominatee among the options rather than using a greedy approach. But the main problem is large message overhead occurred in the network. To understand this approach in brief, consider a network topology as shown in Fig. 4a. Suppose A_2 node is failed. Then, the closest neighbor will be A_3 if the distance between A_2 and A_3, denoted as $|A_2A_3|$, is the minimum among all the other links of A_2. Once A_2 determines its closest neighbor, it sends a message to A_3 and designates it as the node to handle its failure. When it actually fails, A_3 will pick A_5 and A_5 will find A_8 as the closest dominatee to A_3 as shown in Fig. 4b.

6.1.4 Actor to Actor Connectivity Restoration Algorithm (AOM)

Azadeh Zamanifar et al. [25] proposed more efficient distributed hybrid approach to restore the connectivity between actor nodes upon failure of any actor node. AOM has used Stojmenovic's method [26] with connected dominating set (CDS) to find the critical nodes in the network with less pre-failure as well as post-failure message overhead on the network. The authors have considered two steps in AOM (i) Network setup or initialization phase in which two sub-phases are given, (a) Determining the critical nodes and connected dominating set, (b) Determining the restoration policy for each critical actor node. (ii) Recovery phase.

To understand this approach, let us assume a connected topology as shown in Fig. 5.

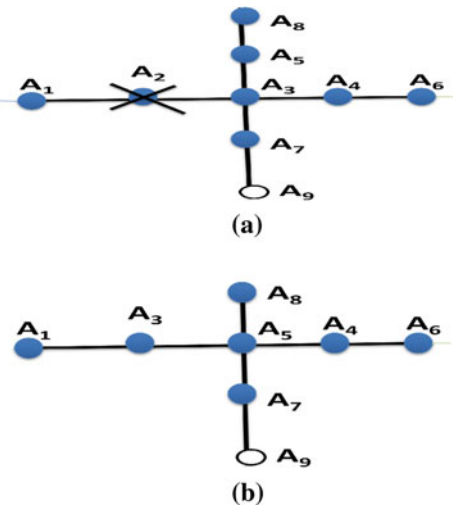

Fig. 4 **a** Network topology with A_2 failed node. **b** Network topology after A_3 will start the recovery process and recovery made by A_5, A_8

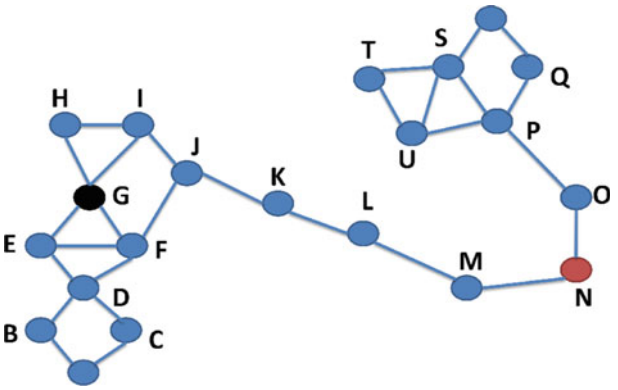

Fig. 5 Connected networks having cut vertices

By using 2-hop neighbors information, two sub-graphs $\{O, P\}$ and $\{M, L\}$ are determined for node N which are two disjoint sub-graphs; thus node N is critical. With the aid of 2-hop neighbors' information, node G is identified as noncritical. Using 2-hop information, it can also calculate CDS of the network (dominator nodes). Therefore, the nodes are categorized as follows:

- Dominatee nodes that are detected as none cut-vertex node by the algorithm like node A as shown in Fig. 5. The movement or failure of these nodes does not partition the network.
- Dominator nodes that are detected as none cut-vertex node by algorithm like C node. The movement of these nodes may partition the network only if the cut-vertex node in their neighbors fails and they relocate.
- Dominator nodes that are detected as the cut-vertex node like K in Fig. 5. The failure of these nodes causes network partitioning.

Jorgic et al. [26] states that the number of actors that are falsely declared as critical depends on the number of neighbors. Since the number of actors in the network is limited, with an average number of 10 to 25 neighbors, the average percentage of actors that are falsely detected as critical would be below 15 %.

After critical nodes in the network are determined, each critical node requests its neighbors to send the maximum distance that they can move towards the critical node without being disconnected from their other neighbors. If the cut-vertex node fails, the network is partitioned into two or more sub-networks. By applying the proposed method, it can determine to which partition each neighbor of the cut-vertex node belongs. If there is a dominatee node (v) in the neighbor of the cut-vertex, a message is sent to v to notify that it must replace the cut-vertex in case of failure. If there is any such actor, the maximum allowable movement of each neighbor of the cut-vertex is calculated.

It must be noted that some neighbors of the cut-vertex may belong to one partition. Therefore, from each partition, the nearest neighbors to cut-vertex are selected and a request message is sent to each of them to send their maximum movement. Each neighbor of the cut-vertex that receives the message calculates the location of the furthest actor in its own neighbor. Therefore, it would determine how much each neighbor of the cut-vertex could move towards the cut-vertex without violating the connectivity of the network. For example, Fig. 6 shows how the maximum movement of each actor is calculated. In order to calculate the maximum allowable movement of A_{11}, in each partition of its neighbors $\{\{A_{13}, A_{14}\}, A_{12}\}$ the nearest actor to A_{11} is selected $\{A_{13}, A_{12}\}$. Then, the farthest actor among them A_{12} is selected. Each actor neighboring the cut-vertex calculates t, the maximum allowable distance as: $t = (r - d') \times \cos(\emptyset)$, where $r - d'$ is the distance that A_{11} could move in order not to be disconnected from its farthest neighbor A_{12}. At the critical node, after receiving all the messages from neighbors, it checks whether there is a neighbor v, who's *MaxMovement* covers the location of the cut-vertex. This condition occurs if v does not have any other neighbor except the cut-vertex, so it is a good candidate for replacement. If this criterion is held, v is responsible for handling the failure of the actor and it would replace the cut-vertex in case of failure. If this criterion is not held, the cut-vertex calculates whether the network could be reconnected if neighboring actors move by their *MaxMovement*, (detailed description of

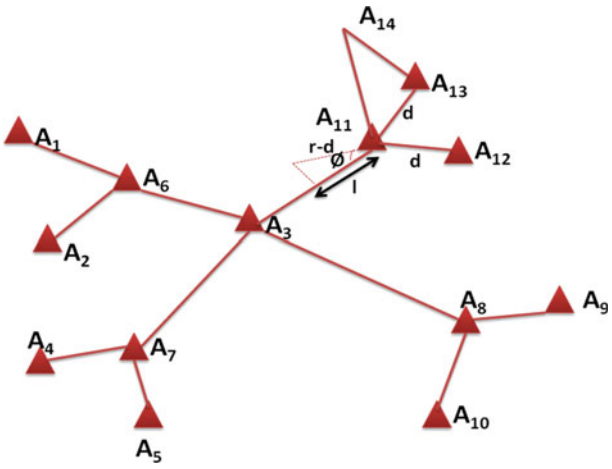

Fig. 6 Maximum allowable movement of an actor node

Fig. 7 Packet format

Node_Id	Relative Position

pseudo code is given in base paper of authors). In Fig. 6, if A_3 fails, A_7 is responsible for finding a nearest dominatee actor A_4 to replace the failed actor. The network is then restored by cascaded movement, meaning that A_7 is replaced by A_4 and A_3 is replaced by A_7.

AOM is efficient for recovery of single node failure as claimed by the authors, but it has some drawbacks:

- AOM has not given the clear explanation, how the failure handler (FH) will perform the recovery after the single nodes' failure.
- No convergence point is mentioned.

6.1.5 Recovery Through Inward Motion (RIM)

Mohamed et al. [27] depicted a localized distributed algorithm for network partitioning recovery through network inward motion. The objective is to reduce the distance travelled by individual nodes during the recovery process. The main idea used by the authors to move the neighbors of a failed node inward toward the position of failed node so that they would be able to reach each other. The approach only requires the knowledge of 1-hop neighbors to recover from a failure. For maintaining a list of 1-hop neighbors, RIM requires only 1-hop neighbors with their relative position and proximity. Each entry in the table consist of two parameters (a) Node_Id, a unique identification assigned to each node in the network. (b) Relative position, Euclidean distance with respect to its neighbors as shown in Fig. 7.

RIM has two steps:

- *Detecting a failure and initiating the recovery process*: Node will periodically send the heartbeat messages to their neighbors to ensure that they are functional. Missing heartbeat messages can be used to detect the failure of a node. Depending on the position of failed sensor node, the impact of failure can be significant or minor. However, the underline philosophy for RIM is to avoid the analysis of whether the absent of failed node causes a network partitioning or not. Instead the recovery is applied for failure of both cut-vertices and ordinary nodes.
- *Cascaded node relocation*: As neighbor of failed node would initiate the restoration process with their individual uncoordinated motion failure node (S_f). Every node (S_a) $\in$ Neighbors of failed node (S_f) send a notification message to all its 1-hop neighbors to alert and states that where it will locate. In addition, S_a informs its neighbors about its rank relative to failed node which is equal to 1 for all neighbors.

The authors considered nodes' rank for cascaded movement from outward to inward. The cascaded movement terminates when there will be no more neighbors disconnected or upon reaching the network periphery. To understand the algorithm, let us consider a network topology having node F fails as shown in Fig. 8a.

In Fig. 8a, A, B, G and H detect the failure of node F. Firstly, each node notifies the neighbors who will lose their connection to it when it moves. G and H have no other neighbors, while A and B notify $\{C\}$ and $\{C, D\}$, respectively. Then, nodes A, B, G and H move directly toward F and become connected, as seen in Fig. 8b. Since C receives two notification messages from A and B, it identifies a new position at the intersection of circles centered at A and B. Node C notifies node D, which will be disconnected when C departs, then moves as depicted in Fig. 8c. Node D also performs cascaded motion, but it determines

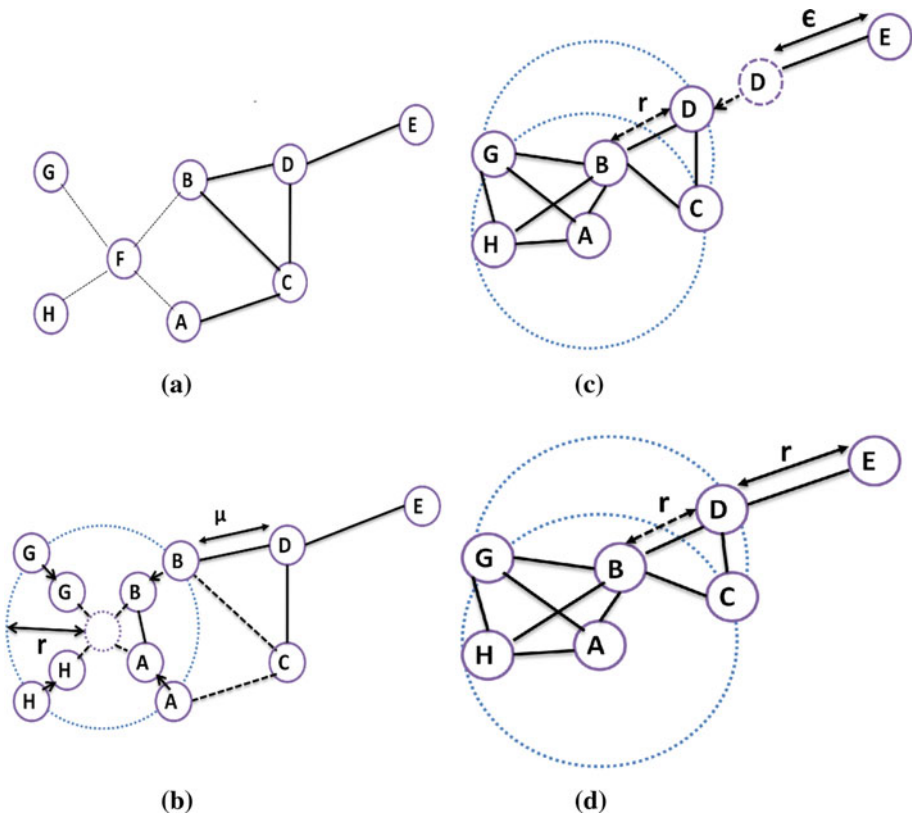

Fig. 8 **a** Network with node F fails. **b**, **c**, **d** Network recovery with the movements of Node *B, A, G, H* with RIM approach

its new location based only on B's position. It ignores C's message since its rank is two whereas the rank of B is one. Nonetheless, when D arrives at its new position, it reconnects with C (Fig. 8c). It is worth noting that the distance between D and B is now r, rather than μ, with $r < \mu$. Node D will notify E prior to moving. Finally, E travels and settles r units away from D, where $r \geq \varepsilon$. The restoration process ends when E settles as shown in Fig. 8d; the important point is that each node moves only once.

In nearest neighbor approach (NN), Nodes A, G and H are aware that node B is the nearest to F and will wait for B to move. Node B notifies C and D before it starts moving and A, G and H will notify after reaching the new position. Figure 9b, c shows the topology after B replaces D and F replaces B. E will be only affected node and will move to position where D was as shown in Fig. 9d.

Although, RIM has reduced the messages overhead due to maintaining only 1-hop neighbors' information and large movement of nodes, it has increased the number of relocated nodes for recovery and disrupts network topology a lot.

6.1.6 Connectivity with Application Level Constraint on Actor Mobility (C^2AM)

Abbasi et al. [28] proposed a self-healing, lightweight and localized distributed recovery algorithm that includes the application level tasks constraints on the actors' mobility while

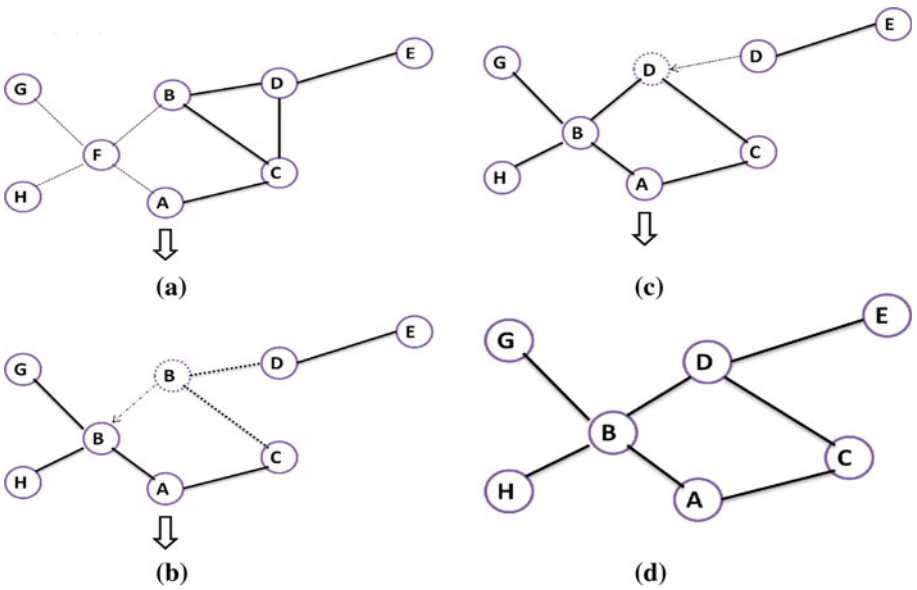

Fig. 9 The changed topologies after recovery of failed node *F* using Nearest Neighbors approach (NN)

restoring the network connectivity. The idea is to take application level constraint on actor mobility based upon certain emergency. In such type of scenario, the authors considered some natural disaster like earth quake or hurricane and used UMRV (unmanned robotic vehicles) equipped with sensors for helping the victims. The job of these sensors is to probe the existence of human live being in the vicinity and report it to the actors. After receiving such type of report, the nearby actors are responsible to reach the location and provide necessary life support until the rescue team arrives. Therefore, at the time when an actor is busy in providing emergency help to a survivor under rubbles, task termination and mobility of this unit may cause serious damage to the operation. However, after completing the operation, the unit can be mobilized to any location without constraints. For constrained mobility, the authors have used mobility index (MRI) value for each actor based on its availability to move and mobility potential index (MP) value to calculate the number of neighboring actor which it can moves. This MRI is then used to decide whether an actor is involved in the connectivity restoration process or not. Now, every actor in this approach would maintain MRI and MP in the range (0-l). MRI is entirely based on the importance of current task. A MRI zero means the actor is free; while a value one means the actor cannot move. In addition to MRI, each node would maintain MP by tracking its 1-hops neighbors and MRI has highest priority over MP. C^2AM approach has three steps:

(i) *Maintaining a list of 2-hop Neighbors*: C^2AM requires every actor to maintain an updated list of its neighbors. To keep the scope of the recovery local, actors store information about 1-hop and 2-hop neighbors only. To keep the list up to date, an actor will send heartbeat messages periodically to update neighborhood information to its reachable actors and to assure them about its proper operation. Each entry in the *TwoHopTable* contains five tuple {*Node_Id*, *MRI*, *MP*, *Node Degree*, *Relative position*}, where *Node_Id* is a unique identifier for an actor at the network level. The information stored in *TwoHopTable* is critical for the successful network recovery since it allows a node to know which actor

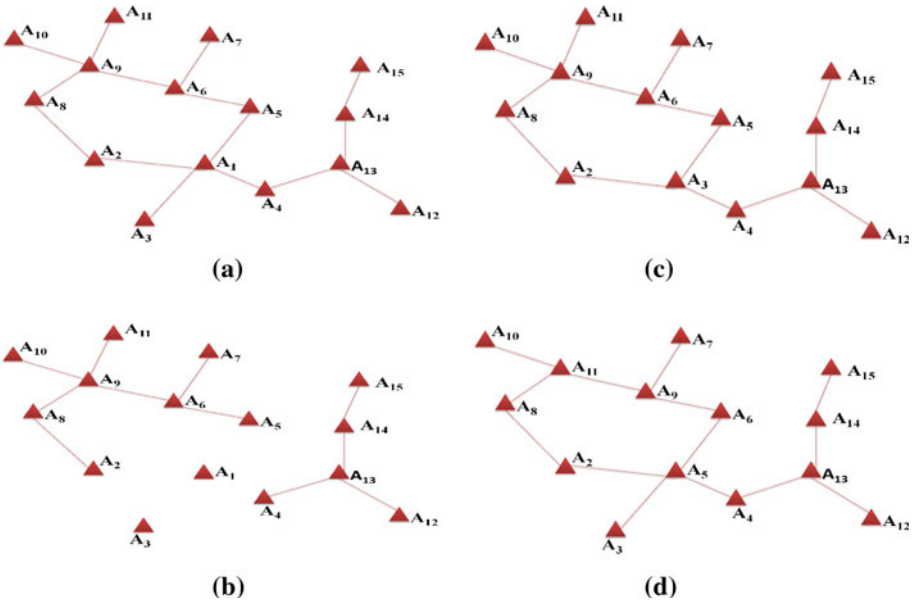

Fig. 10 **a** Connected network topology. **b** Network topology after A_1 failure, contains three disjoint sub-networks. **c** Connected network topology after A_1 is replaced by A_3. **d** Connected network topology after running C^2AM with node A_5 replacing A_1, followed with cascaded movements of A_6, A_9 and A_{11}

is the most qualified to perform the recovery. A node that passes the qualification test would be considered as the most suitable replacement of the failed node. The authors have considered it $Apassed$. The *TwoHopTable* would be updated immediately after $Apassed$ has reached to its new location. In addition, an actor that intends to change its position will inform its neighbors beforehand in order to avoid being wrongfully perceived as faulty. In addition, it would inform its new 1-hop neighbors by broadcasting a *HELLO* message as soon as it arrives at its new location.

(ii) *Detecting a Failure and Initiating the Recovery Process*: To detect a failure, C^2AM watches for repeated misses of the heartbeat messages in order to avoid overreacting to occasional packet losses over the wireless medium and to make sure that all neighbors of the failed node has a consistent assessment about failed actor. The failed actor is referred as A_f. Execution of C^2 AM will be triggered only if a critical node, i.e., cut-vertex, has failed. The *TwoHopTable* will be used to identify cut-vertices in the network.

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

(a) Least MRI Node (b) Highest MP value (c) Least Node Degree (d) Closest proximity to failed Actor.

To understand this approach, let us consider a network topology shown in Fig. 10a by assuming some attributes values stored in Table 1.

Table 1 Attributes table of actor nodes shown in Fig. 10a before the failure of any node

Node ID	MRI	MP	Node degree
A_2	5	1	2
A_3	5	0	1
A_4	3	1	2
A_5	1	0	2
A_6	5	3	2
A_7	3	0	1
A_8	4	1	2
A_9	3	3	3
A_{10}	3	1	1
A_{11}	2	1	3
A_{12}	5	0	1
A_{13}	5	1	3
A_{14}	5	0	2
A_{15}	5	0	1

C^2AM looks first for the node with the least MRI among the 1-hop neighbors of the failed node. Note that among the 1-hop neighbors of A_1 only actor A_5 has least MRI which is 1. Thus, A_5 qualifies for replacing A_1. Since actor A_6 is the only child of A_5, it will move to the location of A_5 despite the fact that it has MRI of 5. A_7 and A_9 are children of A_6 and both have same MRI value of 3, thus A_9 with the higher MP value qualifies to move to the location of A_6. Among the children of A_9, A11 qualifies for moving since it has a MRI value of 2 that is lower than that of A_8 and A_{10}. Since A_{11} is a boundary node and has no children, the restoration process will terminate. Figure 10b–d illustrates the network after successful recovery. It is worth noting that if A_4 has MRI of zero it would be selected as a replacement of A_1. Since A_{13} is the only child of A_4, it simply would move to the location of A_4. MRI and MP values of A_{12} and A_{14} are similar; therefore the node degree breaks the tie and A_{12} replaces A_{13}.

C^2AM is good for such type of applications where mobility over application level constraint is mandatory, this approach has following short comes:

- C^2AM considered one impractical assumption that most of the time there would be some available actors with MRI less than threshold value in the neighborhood of a failed actor which can participate in the recovery process, but this would not be true for maximum cases.
- No criteria is seen how to calculate cut-vertex and non cut-vertex. It is just assumed in the approach.
- Message complexity is higher due to caring for actor's involvement.

6.1.7 Volunteer-Instigated Connectivity Restoration Algorithm (VCR)

Imran et al. [29] illustrated a distributed algorithm called Volunteer-Instigated Connectivity Restoration Algorithm (VCR) to repair served connectivity while imposing minimal overhead on the nodes. The idea is to choose the neighbors of a failed actor as volunteer to restore the connectivity by exploiting their partially utilized transmission range and by repositioning closer to the failed actor. The rationale of VCR is based on instinctive social behavior that can

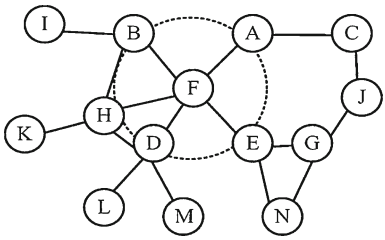

Fig. 11 Connected networks

be observed frequently in most of creatures. For example, in case of a persons death in our social environment, the most closely concerned peoples among many acquaintances examine their availability based on their commitments and voluntarily take up the responsibilities in addition to their own responsibilities. Similarly, in VCR the failure of any node is detected by immediate neighbor actors, because they are directly affected. These affected actors examine their proximity to failure actor node and partially utilized their transmission range in order to decide whether to participate in the partitioning recovery or not. The actors which will participate in the process are called volunteer actors (VAs). The volunteer actors jointly take up the responsibility in order to restore lost connectivity. The actor nodes that could not help in recovery process due to some unfavorable environment conditions are called horrid actors (HA).

VCR contains two steps for recovery process. In the first phase, volunteer actors are identified. In the second phase, the topology repair is performed through uncoordinated relocation of the volunteer actors while exploiting partially utilized transmission range and actor diffusion in the network. To understand the approach, let us consider a connected network topology as shown in Fig. 11.

In this topology, node F is supposed to fail, an actor A ∈ Neighbors (F) calculates its distance d (A, F) to F. If d (A, F) is more than $\alpha\, r_{max}$, actor A is not required to participate in the recovery at this time, i.e., close neighbors to F are favored as volunteers. Assume uniform node placement of n nodes in a square area (LxL). The distance between two nodes in the same row is $\left(\frac{L}{\sqrt{n}}\right)$ and the distance between two diagonally neighboring nodes is $\left(L\frac{\sqrt{2}}{\sqrt{n}}\right)$. Therefore, the initial value of α is set as the average proximity to neighbors: $\alpha = (0.5)\left(\left(\frac{L}{\sqrt{n}}\right) + \left(L\frac{\sqrt{2}}{\sqrt{n}}\right)\right)$. It is nothing that α is increased, if actor A is not connected within a preset time in order to increase the threshold for not participating in the recovery. In other words, an actor node can switch from horrid state to volunteer state depending on the observed progress on restoring connectivity.

VCR is a good approach with little number of nodes movements, but main problem with radio signal interference between the nodes. With spreading of volunteers increase the transmission power of volunteer actor and enables it to reach other neighbors of failure actor as well as its children while and after moving. Volunteer actors may negatively affect other node in the vicinity. In particular, if the network gets partitioned increasing the transmission power of volunteers from the same segment after they move close to failed node will boost medium access contention and radio signal interference.

6.1.8 Least Disruptive Topology Repair Algorithm (LeDiR)

Abbasi et al. [30] suggested a localized and distributed algorithm called LeDiR for repairing of inter-actor topology with least number of nodes movements without imposing additional pre-failure communication overhead. The idea is to take the local view of a node about the

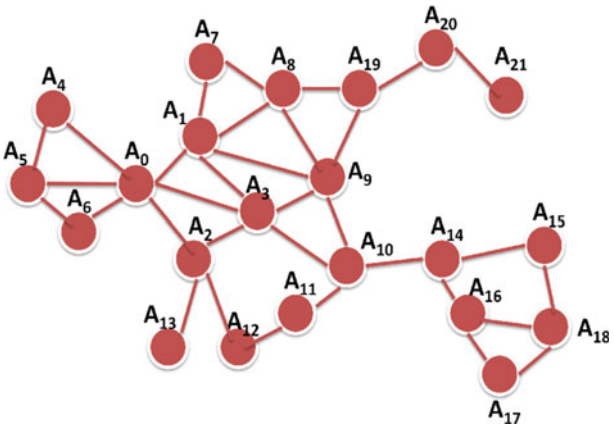

Fig. 12 Connected network with A_0, A_{10}, A_{14}, and A_{19} are cut-vertices whose failure leaves the network partitioned into two or multiple disjoint blocks

network to relocate the least number of nodes with block movements instead of individual nodes in cascaded movement and ensure that no path between any pair of affected nodes is extended relative to its pre-failure status. To understand this approach, let us assumed a network topology as shown in Fig. 12.

This approach has four major steps:

(i) *Failure Detection*: Once a failure is detected in the neighborhood, 1-hop neighbors of failed actor would determine the impact, i.e., whether the failed node is a cut-vertex. This can be calculated using the shortest routing table (SRT) maintained by each node in the network.

(ii) *Smallest Block Identification*: LeDiR limits the relocation to nodes in the smallest disjoint block in order to reduce the recovery overhead. The smallest block would be identified by finding the reachable set of nodes for every 1-hop neighbor of the failed node after the failure and then picking the one with the fewest nodes. Since a cut-vertex will be on the shortest path of two nodes in separate blocks, the set of reachable nodes can be identified through the use of the SRT, after excluding the failed node. In other words, two nodes will be connected only if they are in the same block.

(iii) *Replacing Faulty Node*: If node X is the neighbor of the failed node that belongs to the smallest block, X is considered the best candidate to replace the faulty node. The reason is that moving a node and its children from the smallest block would most probably yield the least total travel distance, if the entire block has to move. In case more than one actor with such characteristics exists, the closest actor to the faulty node would be picked first. Any further ties will be resolved by selecting the actor with least Node_Id.

(iv) *Children Movement*: When node X moves to replace the faulty node, possibly some of its children will lose direct links to it. In general, it should not happen since some data paths may be extended. Actually, in Fig. 12, the path between A_2 and A_3 got extended because A_2 lost its link to A_{12} after A_{12} has moved. LeDiR opts to avoid that by maintaining the existing links. Thus, if a child receives a message that the parent is moving then the child would notify its neighbors and travel until reconnecting with the parent again. If a child receives notifications from multiple parents it would find a location from where it can maintain connectivity to all its parent nodes.

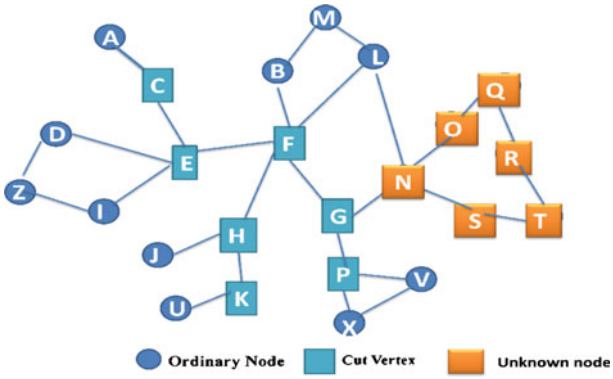

Fig. 13 Connected networks

To understand the working of LeDiR, Suppose a node A_{10} fails in the topology as shown in Fig. 12. In this case, LeDiR will only involve the block of node A_{14} for movements. In addition, LeDiR opts to avoid the effect of the relocation on coverage and also limits the travel distance by stretching the links and moving a node only when it becomes unreachable to their neighbor.

LeDiR is efficient approach for delay-sensitivity application where path extended between the nodes after the recovery from the network partitioning may give the negative effect or even make the application useless. LeDiR has some drawbacks:

- LeDiR does not provide any mechanism to detect the cut-vertices. It is assumed that this information is available at the node which may require the complete knowledge of whole topology.
- LeDiR has decreased the MMI of all involved actor nodes, but increased the TMI of all involved actor nodes which may increase the message overhead on the network.
- LeDiR move the complete block for recovery means number of moving nodes is more and energy consumption is also more.

6.1.9 Coordinated-Assisted Connectivity Recovery Approach (CCRA)

Zhao et al. [31] demonstrated a new distributed scheme called Coordinated-Assisted Connectivity Recovery Approach (CCRA) to handle the network portioning problem with least number of nodes and least movements of nodes. The idea proposed in CCRA is not to calculate cut-vertices and non cut-vertices in advance at pre-failure, but calculate at the time of node failure to reduce the pre-failure message overhead on the network. In CCRA, the actor nodes are behaved into three roles namely ordinary node (leaf node), cut-vertex and unknown nodes denoted by n_0, n_c and n_u respectively. If a node cannot determine whether it is n_c or not by local information, then it is n_u. The goal of this approach is to minimize the number of n_u by coordination between actor nodes, therefore to reduce the useless movement for connectivity during recovery. To understand this approach, let us consider a network topology as shown in Fig. 13.

CCRA uses three predicate rules to find the class of actor nodes.

Rule 1: If $|N1(u)| = 1$, then u is leaf node (n_o) and the node $vvN1(u)$ is n_c.

Where $|N1(u)|$ is number of neighbors of u and $N1(u)$ is 1-hop neighbors of u

Rule 2:$\forall v, w \in$N1(u) $\exists$ *path* $(v \ldots x \ldots w) \wedge x \neq u$ then u is *n*o.

Rule 3:$\exists v \in N1(u) \wedge \quad |N1(v)| > 1 \wedge cf(v) = |N1(v) - u|$ u is n_c when $|N1(u)| > 1$

Where $cf(u)$ is crux factor which is defined as if some neighbors of u can help u to confirm its type then crux factor of u is s which is denoted by $cf(u) = s$.

Clearly, nodes A, U and J are n_0 and nodes C, K, H are $n_c$according to rule 1. The node D also is n_0 by rule 2 because all of its 1-hop neighbors can visit each other within 2-hop. Similar to D, nodes Z, I, X, V, B, L and M are n_0. With the help of the nodes X and V, node P knows that it is n_c and$cf(P) = 2$. After that, node G can deduce it is n_c and $cf(G) = 1$ because only the neighbor P satisfy the rule 3. Apparently, node E is n_c with help of node C by rule 3. In 2-hop information, neighbors D and I cannot help E to determine its role, therefore $cf(E) = 1$. Node H is n_c can be inferred by J or K, so $cf(H) = 2$. Similarly, node F is n_c with help of neighbor H. The rest of actors are n_u. After selecting the node class and type of an actor failure in the network, CCRA use branch cost and node cost for the recovery process. (For complete details refer the base paper).

CCRA approach is efficient for distributed single node failure recovery through actor coordination, but one of the main problem is extra message overhead exists for coordination among the actors to determine the cut-vertices which reduces network lifetime.

6.1.10 Least Movement Topology Repair Algorithm (LeMoToR)

Abbasi et al. [32] described a localized and distributed scheme to deal with network partitioning with least number of nodes have to move with less message overhead on the network. The idea of LeMoToR is to utilize the existing path discovery activities at restoration time in order to know the structure of topology and take appropriate action accordingly. LeMoToR relies on the local view of a node about the network to start recovery process. It applies recursively on every node of a particular path to sustain the intra-smallest block connectivity. To understand the complete approach, consider a network topology as shown in Fig. 12.

The main idea of LeMoToR is to replace the faulty node by selecting a neighbor node that belongs to the smallest disjoint block. In this case, if neighbors of selected node get disconnected from its parent node within the block, LeMoToR is further applied recursively. This will not only move the least number of actor nodes but also limit the recover overhead in terms of distance that the nodes collectively travels. From Fig. 14a, if node A_{10} fails, LeMoToR only involve the block of node A_{14}, because this block is smallest block among all the disjoint blocks. In addition, LeMoToR limits the travel distance by applying itself recursively and moving a node only when it becomes unreachable to their neighbors. This approach has mainly four steps:

(i) *Failure Detection*: To detect a node failure in the neighborhood, an exchange of heartbeat messages is assumed in the network. After n missing heartbeats, a node F would be assumed faulty. If the failed node F is a cut-vertex, network recovery would be triggered on the 1-hop neighbors of F.

(ii) *Smallest Block Identification*: After a cut-vertex failure, 1-connected network G is split into more than one connected component, i.e., sub-networks. Each sub-network consists of few nodes of G that are 1-connected to each other within the sub-network. Basically, each sub-network is a separate "*block*" that was connected to the other blocks in G via faulty cut-vertex. LeMoToR attempts to find a block among the disjoint blocks that consists of the least number of nodes, referred as "*smallest block*". Actually, LeMoToR aims to confine the node movement within the smallest block to minimize the node movement. To identify the smallest block, every 1-hop neighbor of faulty node would

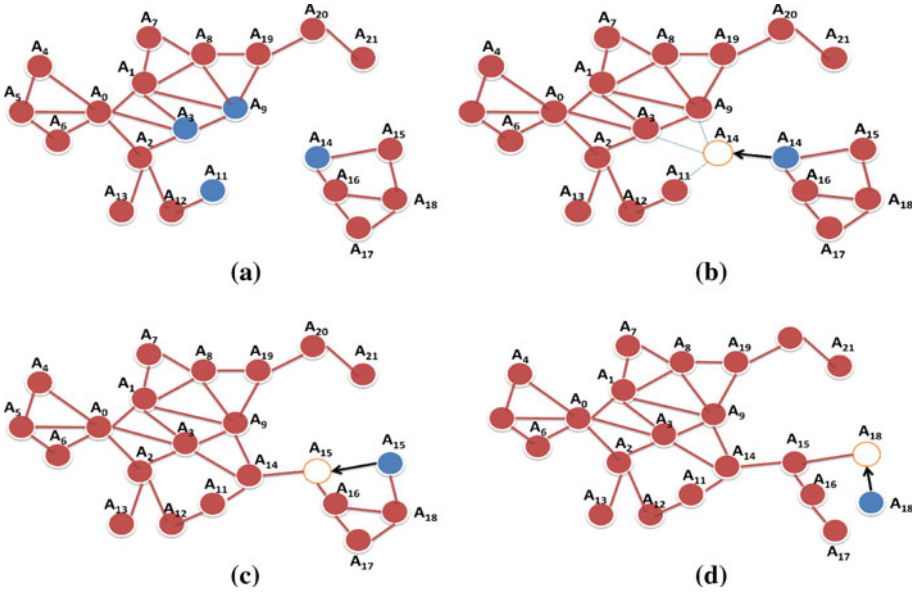

Fig. 14 Example showing how LeMoToR restores the connectivity after the failure of A_{10} node

identify the reachable set of nodes for itself and every other 1-hop neighbor of the failed node by using SRT. The block with the fewest nodes is identified as a smallest block.

(iii) *Replacing the Faulty Node*: To replace the faulty node F, a neighbor node J is selected from the smallest block. The reason is that LeMoToR strives to minimize the number of node movements during the network recovery. Since LeMoToR is recursive in nature, moving a node and its children from the smallest block would most probably involve the fewest actor nodes in the recovery. In case more than one actor with such characteristics exists, the closest actor to the faulty node would be picked. Any further ties resolve with Node_Id.

(iv) *Children Movement*: When node J moves to replace the faulty node, possibly some of its children nodes will become disconnected. To regain the connectivity, children would assume the moved parent node as a dead node and would apply LeMoToR at the children level. The smallest block at the children level would be identified. The child that belongs to the smallest block would proceed to the location of already moved parent node. This phenomenon would continue until all the nodes are reconnected with the network G. To understand this approach, again, if A_{10} node fails as shown in Fig. 14a, A_{14} becomes its neighbor that belongs to smallest block. Node A_{14} notifies its neighbors and moves to the position of A_{10} to restore connectivity as shown in Fig. 14b. Disconnected children, nodes A_{15} and A_{16}, execute LeMoToR again to find out which one of them should move to the location of A_{14} Obviously, node A_{15} belongs to the smallest block and thus moves to the location of A_{14} to maintain the communication link (Fig. 14c). Note that the reason to execute LeMoToR recursively and to identify the smallest block even among the children movement is to minimize the overall node movement. Nodes A_{15} would notify its only child A_{18}, before it moves. Since A_{18} is the only child, it simply belongs to the smallest block and moves to location of A_{15}. Figure 14d shows the repaired network after the recovery for A_{10} node.

LeMoToR is efficient real time restoration approach in terms of number of nodes to move and distance traveled by the nodes, this approach has following drawbacks:

- In LeMoToR, lot of computation is required on every node, because of use of recursive process to find the best route of recovery.
- The approach has not cleared how to find smallest block either using greedy approach or depth first approach among the disjoint blocks. For searching a smallest block, it generates lot of communication overhead.

6.1.11 Least Distance Movement Recovery Algorithm (LDMR)

Alfadhly et al. [33] proposed a distributed algorithm called Least Distance Movement Recovery Algorithm (LDMR), which exploits non-cut vertices nodes for the recovery so that no further partitioning occurs after the movement of these nodes. The idea is to move the direct neighbors of failed node position while its original position is replaced with the nearest non cut-vertices. LDMR exploits node mobility and the availability of non cut-vertices in the network in order to minimize the distance that nodes collectively traveled during the recovery process. The distinct feature of LDMR is the avoidance of the cascaded movement spread throughout the whole network**.** To understand the complete approach, let us consider a network topology as shown in Fig. 12.

LDMR has five steps for recovery:

(i) If an actor X is damaged or stops functioning, e.g. due to battery exhaustion, for example, this failure is detected by the neighbors of X denoted by $\mathrm{A_{Ni}}$ due to the absence of the heartbeat messages which should have been sent by X periodically.

(ii) Each neighbor in step 1 and not within $r/2$ distance from X starts a search process looking for the nearest non cut-vertex node, where r is the communication range. This non cut-vertex is called a candidate node (C_{ij}). The neighboring nodes (A_{Ni}) of failed node broadcast a search message containing several entries such as failed node ID, neighbor node ID and, Time-To-Live (TTL).Then, the nearest non cut-vertex node replies to this message with its distance to its parent node. Each neighbor chooses the best candidate among received responses based on the distance D_i from the failed node.

(iii) Then, A_{Ni} sends a request message commanding C_{ij} to move to its position. Upon receiving this message, the commanded node acknowledges this message and starts moving to the specified position. This acknowledgment is necessary to avoid choosing the same non cut-vertex node by more than one neighboring node. Therefore, the commanding node $\mathrm{A_{Ni}}$ should wait for the acknowledgment before moving. If a node does not receive an acknowledgment, it selects the next nearest candidate and so on. It is worth mentioning that potential candidates, including C_{ij}, will query its 1-hop and 2-hop neighbors and apply the CDS algorithm in order to know whether is not a cut-vertex, and is able to declare its candidacy and respond positively to the request.

(iv) Each neighbor node $\mathrm{A_{Ni}}$ moves toward the position of X until it becomes $r/2$ away of it. If one of the neighbors is within this distance, no need to move further. Each candidate node C_{ij} sends movement notification message to its neighbors before sending the acknowledgment to the direct neighbor requestor $\mathrm{A_{Ni}}$. This notification is essential to avoid network partitioning that may occur when multiple non cut-vertices neighbors move simultaneously. If other neighbors receive similar requests to move, then, the nodes that believe that this movement may partition the network, they send panic messages to prevent this movement.

(v) After the movements in step 3 and 4, the network connectivity should have been re-established.

In Fig. 12, after the failure of A_{10}, its direct neighbors $\{A_3, A_9, A_{11}, A_{14}\}$ detect the failure and start the recovery process by searching for the nearest non cut-vertex nodes. The search process may consume lot of communication messages which is not desirable in a constrained environment such as WSAN. Therefore, each node broadcasts a search request message and includes a Time-To-Live (TTL) parameter. In this example, we assume nodes $\{A_3, A_9, A_{11}, A_{14}\}$ starts with TTL equals 3 as shown in Fig. 15a. Each receiving node of this message decrements the TTL value and forward the message if the TTL is still greater than zero. If the receiving node is a non cut-vertex, it will discard the request unless it comes from another initiator, i.e., neighbor of A_{10}. Nodes $\{A_1, A_4, A_6,$ and $A_{13}\}$ respond to the request of node A_3, while nodes A_{15} and A_{16} respond to the request of A_{14}. In addition, nodes A_8 and A_{21} respond to the request of A_9, and node A_{12} responds to the request of A_{11}. Based on the distance between the potential candidate and the neighbor node that initiates the request, the closest candidate is picked and notified. Each candidate sends an acknowledgment to the corresponding node and starts moving. After receiving the acknowledgments, nodes A_3, A_9, A_{11}, and A_{14} move toward the position of the failed node A_{10} until they are $r/2$ away from A_{10}. These movements ensure all nodes are connected to each other as shown in Fig. 15d. Candidate nodes also inform their neighbors before sending the acknowledgment messages and then wait for some time to check the response of its neighbors. If no message is received, the Candidate node C_{ij} sends the acknowledgement message to the requester (R_{ij}). In certain scenarios, more than one neighboring non cut-vertex nodes may move simultaneously as A_8 and A_1 in above topology. This situation may lead to partitioning the network again (A_7 is disconnected). Let us assume A_8 sends the notification message first, A_7 still is connected to the whole network via A_1. Now, if A_1 sends it notification message, A_7 will send a message which prevents A_1 from sending the acknowledgment message. Consequently, if A_{Ni} does not receive an acknowledgment message from the nearest candidate, it picks the next nearest. In our example, it picks A_4 instead of A_1.

This approach is efficient approach with least distance movement of all involved nodes in the recovery, it has certain drawbacks:

- Message complexity is high due to every neighbor of the failed node will send the request message to its neighbor for non cut-vertex node determination and gets the acknowledgment accordingly.
- Number of involved nodes is larger as compared to other efficient approaches.

6.1.12 Coverage Conscious Connectivity Restoration Algorithm (C^3R)

Neelofer Tamboli et al. [34] addressed first coverage conscious connectivity restoration algorithm called C^3R which integrates coverage with connectivity during partition recovery. C^3R involves multiple neighbor nodes as failure handler to recovery failed node. All failure handlers temporarily relocates on the position of failed node one at a time and return back to its original position to serve their original neighbors. The distinct feature of C^3R is using spare nodes as failure handler to avoid further partitioning due to movement of nodes. To understand the complete approach, consider a network topology as shown in Fig. 16.

C^3R has four steps as follow:

(i) *Startup operation:* In this approach, every node maintains 1-hop neighbor information during bootstrap stage with broadcasting $HELLO$ messages to its neighbors. Each node

tabulates relative position and ID of all its neighbors. This list is updated dynamically with nodes periodically sending $HEARTBEAT$ messages to their neighbors. When a node does not receive any response to its $HEARTBEAT$ message from a neighboring node it assumes that node has failed.

(ii) *Failure detection:* Upon detection of failure of a node, its failure handler initiates recovery procedure immediately unless it is not participating in recovery process of another node. C^3R applies recursively on each node without considered the failed node status i.e. node is cut-vertex or non cut-vertex. Since it takes coverage with connectivity issue.

(iii) *Planning Recovery process*: After the failure detection and before the recovery starts the failure handler informs its neighbors temporarily about its relocation to failed node to avoid being perceived as failure node. Neighbors that route data through that node (i.e. failure node) would either find new routes in the meantime or buffer the data until that node returns to its original position. In addition, each of the concerned nodes calculates the degree of overlapped coverage, which is the percentage of the total area covered by this node that also lies within the sensing range of at least one of its neighbors.

(iv) *Execute recovery plan*: The recovery process planned by the coordinator defines for each concerned node what to do. Nodes, that are spared, go back to their original position and resume their duties. The role of a participating node is to go to the position of failed node and spend some time there and then comes back to its original position. While on the position of failed, node links all neighbors of failed node to ensure coverage and network connectivity for current session. Before go back to original position, the next node on the given schedule starts moving towards failed node, and so on. When a node returns home, it notifies its original neighbors and resumes routing of their buffered data packets. After all scheduled nodes finish one round then same process repeats again by all similar nodes. Thus the participating nodes are periodically moving back and forth until additional nodes are added to the network.

In Fig. 16 node A2, A7, A9 are neighbors of node A1. The failure of A1would detach A2, A7 and A9 as well as their neighbors from the rest of the network and leave a hole in coverage since no other node has its sensing range overlapping with A1. Although replacing A1with another node will restore the connectivity, it only shifts the coverage hole to another part of the field, either in the inner part of the network or at the periphery. C^3R opt to overcome this problem by temporarily replacing the failed node with one or multiple of its neighbors (as explained earlier). The involved nodes will switch back and forth so that the network topology and the coverage stay mostly the same to their pre-failure status.

Although, this approach is efficient approach with network behavior like pre-failure network, it has following drawback:

- Node movements are more due to more frequent movement of failure handlers forth and back in the network. Therefore, these nodes will consume more energy and have more chances of their failure, and create adverse affect on the network.
- More spare nodes are required in the network, which increases cost of the system.

6.1.13 Node Recovery Through Active Spare Designation (NORAS)

Ketaki Vaidya et al. [35] suggested localized algorithm called Node Recovery through active Spare designation (NORAS) for repairing of inter-node connectivity. The idea is to find spare nodes inside the network prior to failure that can volunteer replace failed node. Identification of spare nodes is done based node node's degree and overlapped coverage area.

NORAS has three main steps:

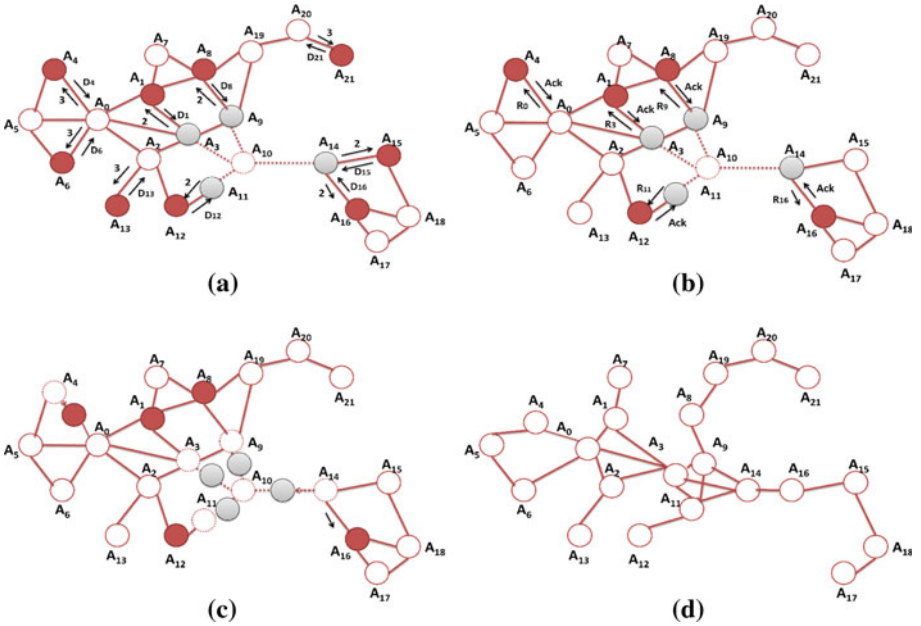

Fig. 15 **a** Node A_{10} fails and its direct neighbors begin the search process, **b** direct neighbors send recovery requests, A_3 send its second request to A_4 since it does not receive ACK from A_9, **c** nodes start moving, **d** final connected networks after recovery

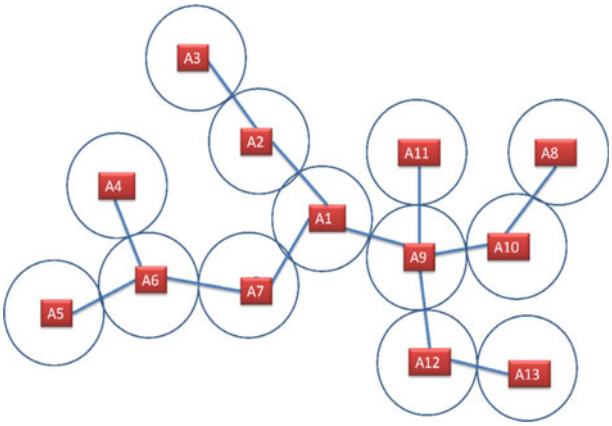

Fig. 16 Connected networks

(i) *Self-assessment:* In this phase, each node performs self-assessment to determine whether it is eligible for Active Spare Nodes (ASN) and can serve as a backup to other failure nodes.

(ii) *Designating Backup:* After estimation of availability, each node informs 1-hop and 2-hop neighbors about its status. A critical node collects the availability of its neighbors and form ANS from which one candidate chooses as backup.

(iii) *Connectivity Restoration*: When the failure of a critical node detects by its direct neighbors, each node checks its candidacy of backup node. If backup is not known, a call for help will be spread out to find suitable candidate. In the worst case scenario, a leaf node will be picked as backup node.

,

NORAS has following drawbacks:

- Due to proactive in nature and storing of 2-hop neighbor information, large communication overhead occurs in the network.
- Large movement of nodes exists as compared with other approaches.

6.1.14 Rapid Spanning Tree Protocol (RSTP)

Kiyoto et al. [36] proposed novel approach called Rapid Spanning Tree Protocol (RSTP) to recovery network failure of complicated topology (i.e. mesh topology) by using tie-sets (Tie-sets are defined as sets of links constitute a loop in the network). The idea is dividing the entire network into small local units which consists of all vertices and links. This approach has four phases:

(i) *Formation of tie-sets*: A tie set is formed based on spanning tree concept (detail is given in base paper). A spanning tree represents a sub-graph which covers all links and vertices in a single network with no loop.
(ii) *Store state information of each vertex*: In this phase each vertex hold three types of state information as given below:

- *Incident Links:* Information of links which are connected.
- *Adjacent Vertices:* Information of vertices which are connected in network**.**
- *Tie-set Information:* Information of tie-set to which network belongs.

(iii) *Failure detection in tie-set*: Tie-set messages are regularly circulating on each tie-set, are used to detect single link failure in network. Figure 17 shows a way to detect link failure in a tie-set. Once failure occurs, the circulating message stops.
(iv) *Recovery based on tie-set*: Each vertex node updates its information of new network based upon new tie-set. In this way recovery occurs in the network.

RSTP has following drawbacks:

- RSTP has longer convergence time in recovery.
- The authors have applied only for ring topology means it is not topology independent.
- Due to flooding of messages in each tie-set creates large communication overhead in the network which is not good for battery constrained devices.

6.1.15 Distributed Connectivity Restoration Algorithm (DCRA)

Zhenqiang et al. [37] suggested a hybrid connectivity restoration framework for mobile multi-agent networks to restore the connectivity of disjoint networks subject to single or multiple simultaneous nodes failures. The authors have addressed two important issues collision avoidance between nodes while recovery process and stability of connectivity. The authors have proposed a novel approach Distributed Connectivity Restoration Algorithm (DCRA) to monitor the neighbor status and select the best available agent(s) to store the connectivity of cut-vertex agent(s). DCRA has two phases for recovery:

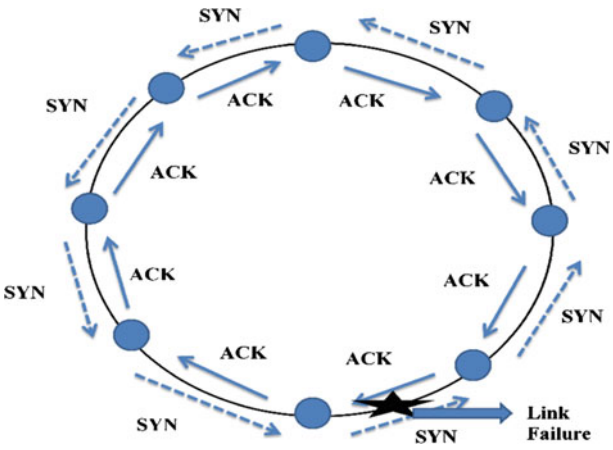

Fig. 17 A tie-set messages in a loop

(i) *Cut-vertex determination and failure detection*: Every agent in the network proactively determines that its 1-hop neighbor is a cut-vertex or not based upon information of its k-hop neighbors. Each agent will periodically send the $HEARTBEAT$ messages to its neighbors that contain information about Node_Id, a flag that indicates whether it is cut-vertex or not, current position, order list and its 1-hop neighbor list. Moreover, to achieve all its k-hop neighbor, the maximum hop count is set TTL = k. The failure of any agent is detected by missing $HEARTBEAT$ massages.

(ii) *Connectivity Restoration*: Based on local determination of cut-vertex agents and failure detection mechanism, DCRA locally selects the best candidate for failure recovery. A major challenge in the recovery process is to further avoid the partition due to movement of nodes. To deal with this issue a disconnected precaution mechanism is developed in DCRA to constraint movements of all agents.

DCRA has following short comes:

- DCRA considered only ring topology for failure recovery which is not practical true for random deployment.
- Message complexity is high due to rapid circulation of synchronous and acknowledgement messages in the network.

6.1.16 Distributed Partitioning Detection and Connectivity Restoration Algorithm and Recovery Algorithm for Multiple Node Failures (DCR and RAM)

Imran et al. [38] have proposed two hybrid distributed approaches for recovery of single node's failure (DCR) as well as multiple nodes' failures (RAM) (i.e. special case of two adjacent simultaneous nodes failure) in WSANs. The idea pursued by the authors is proactively identify critical nodes based on local topological information and designates appropriate, preferably those nearby nodes whose failure(s) does not affect on the network i.e. non-critical nodes. Due to hybrid nature, these protocols are well suited for real-time applications where recovery time must be small to tolerate the bounded delay. The procedure used in DCR is similar to DARA approach with one difference to identify critical nodes using local information based on improved Milenkojogic method [26] (for full detail please refer to base paper).

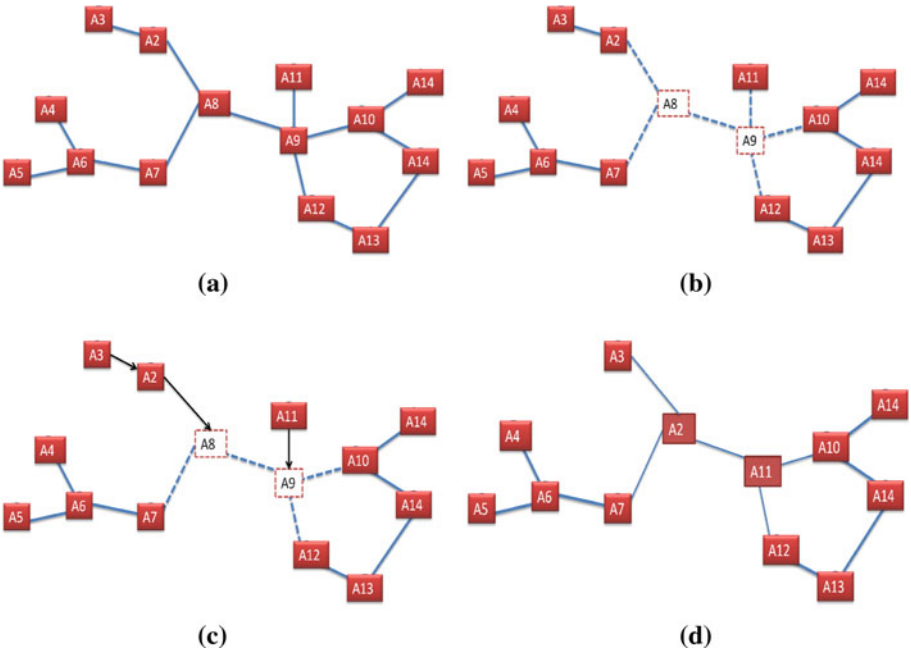

Fig. 18 **a** Connected Networks. **b** Simultaneous failure of Node A8 and A9 in Networks. **c** Recovery process to heal failure of Node A8 and Node A9 by their neighbors A2-A3 with cascaded movement and A11 respectively. **d** Connected Network after recovery of Node A8 and A9

The authors have also proposed second distributed recovery approach called RAM to repair special case of multiple node failure (i.e. two nodes failure). RAM has following procedure to understand the protocol:

Backup Selection and Recovery: Once the critical nodes are identified as per given in Milenkojogic method [26]. The backup nodes are selected based on Neighbor Criticality and Availability (NCR). Basically, each node maintains a state whether it is engaged as backup node for some node or not. This information is periodically sent by all nodes to their neighbors for its engagement. Critical node always chooses a backup node that is not serving another node. In other words, a node could not move more one location as long as another free node is available in the neighborhood. This ensures the recovery in case two adjacent nodes fail simultaneously.

Figure 18a shows the connected network topology in RAM. When simultaneous failure of nodes A8 and A9 occurs in the network, the neighborhood nodes selected from NCR list will start the recovery as shown in Fig. 18b, c. Node A2 is NCR list of A8 and A11 is in NCR list of A9 respectively. As these nodes detect the failure of their base node, they start the recovery. Figure 18d shows the final topology of network after fully restoration of nodes.

6.2 Multi-Node Failures Recovery Approaches

Multi-node failure recovery is more challengeable and hot research area in WSNs now a day. In this section, we explain various multi-node failure recovery approaches based on our literature surveyed work.

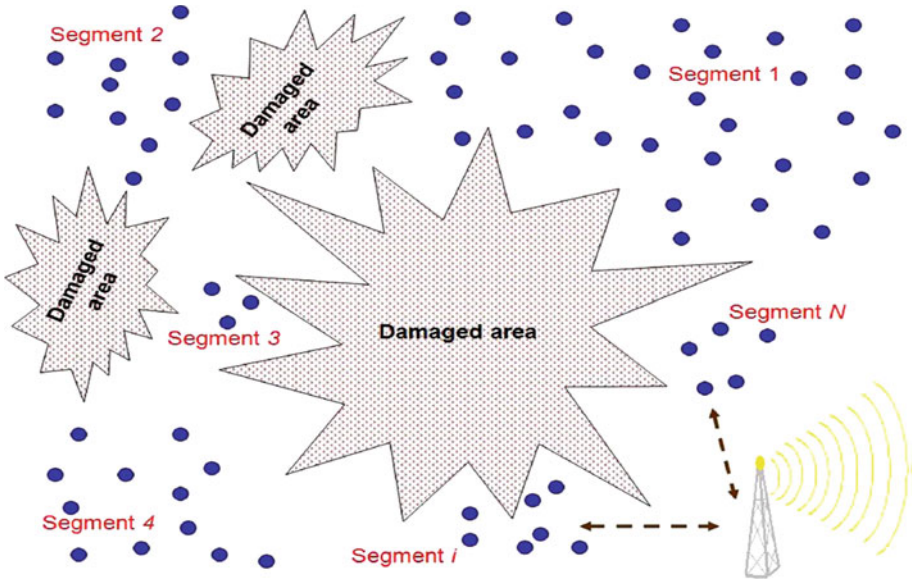

Fig. 19 Partitioned networks [39]

6.2.1 Bio-inspired based approaches (1C-SpiderWeb, 2C-SpiderWeb)

Senel et al. [39] have proposed the first centralized multi-node failure recovery approach for WSNs. The failure of numerous sensor nodes severs the network connectivity by splitting the topology into isolated segments and may thus hinder the proper operation. Therefore, recovering from such a major damage is crucial and more challengeable as compared to single node failure recoveries. Figure 19 shows articulation of damaged network. In order to do recovery in such multi-node failure system, the motivation behind SpiderWeb is planting of relay nodes in damaged isolated area. A relay is a more capable node with significantly more energy reserve and longer communication range than sensors. Although they can, in principle, be equipped with sensor circuitry, relays mainly perform data aggregation and forwarding. Unlike sensors, a relay may be mobile and has some navigation capabilities. Relays are favored in the recovery process, because it is easier to accurately place them relative to sensors and expedites the connectivity restoration among the disjoint segments. Intuitively, relays are more expensive. Therefore, the number of engaged relays should be minimized. It is assumed that all deployed relays have the same communication range r. The distance between every pair of segments may be larger than $2r$, and thus, multi-relay intersegment paths would be necessary.

Problem Statement: Given m disjoint segments of sensors in an area of interest, determine the least count and position of RNs that are needed to connect all segments while maintaining some desirable topology features, such as robustness to relay failure, coverage, and balanced traffic load.

(a) 1C- SpiderWeb Heuristic The idea behind the 1C-SpiderWeb deployment strategy is to place the relays inward to yield better network connectivity and coverage. To balance the intersegment path length in terms of the number of hops, RNs are placed toward the estimated center of mass (CoM) of the segments. Basically, from each partition to the CoM, the system

will gradually deploy nodes until all the partitions are connected. This way, it increases total coverage of the network but reduce the possible number of cut vertices in the network as well. Before the placement of RNs starts, it is necessary to identify the outer segments in the area of interest. To do this, the network randomly picks representative nodes from each partition and run a convex-hull algorithm. The convex-hull algorithm returns a subset of representative nodes that sit on the corners of a convex polygon. After finding the convex polygon, now determine the CoM of the polygon. Nodes are then deployed along the line between a segment and CoM. Obviously, the relays around the CoM will be in the communication range of each other, and the segments thus become connected. Figure 19 shows a network that was split into seven disjoint segments. Running the convex-hull algorithm concludes that $S7$ is an inner segment. Therefore, $S7$ will not be involved in the algorithm. Next, we find the CoM of the polygon whose corners are $S1$, $S2$, $S3$, $S4$, $S5$, and $S6$, where Si denotes a representative node for ith segment. The line between a particular Si and the CoM will be referred to as Li. Depending on the location of the partitions; some relays may become connected before reaching the CoM. The 1C-SpiderWeb algorithm exploits this case to optimize the deployment. Relays are basically populated starting with the furthest partition to increase the probability of reaching an inner partition that may fall in its communication range or another partition on the convex hull either directly or through one of the relays. In fact, as aforementioned, the direction of relay deployment may be changed according to the progress made in the system. The main point is that relays from the various partitions get closer to each other when approaching the CoM. Starting with the furthest partition would increase the probability of reaching one of the existing relays and connecting with another partition, as detailed is given in the following discussion.

Although this case offers an opportunity for minimizing the number of relays, it can prevent all partitions from reaching each other. For example, two segments Si and Sj may be very close to one another and far from the rest. The first relays on the paths (Si, CoM) and (Sj, CoM), respectively, may be reachable to each other, making Si and Sj connected without reaching relays from other partitions.

To avoid premature termination of the relays along a line, the 1C-SpiderWeb algorithm applies a connectivity rule. Before terminating the execution of the 1C-SpiderWeb algorithm, a segment needs to be connected to two neighboring segment: (1) First on to its right and (2) another neighboring segment to its left. Because, it is considered that segments are on the convex hull; every segment will have a neighbor to its right and another segment on its left. Assume that there are m segments on the convex hull. Let Li and Lj be two neighboring lines, where $i = (j - 1) mod m$ (i.e. Lj is the right neighbored Li). If any two nodes on Li and Lj are connected, then Li will be referred to as *right connected*, and Lj is considered *left connected*. If Li is neither left nor right connected, it will be *not connected*. If Li is both left and right connected, it will be *connected*.

Sometimes, one or two partitions may be located very far from the CoM compared to other partitions. In these cases, closer partitions will reach each other in early iterations; however, the algorithm will continue to deploy RNs along the neighboring lines of the furthest partition. To avoid redundant RN deployment, a variable di is introduced, which is the distance between CoM and closest node on Li that decides where to start the RN placement and sort the lines $L1, L2, Lm \ldots$ in descending order according to their di values. Let us assume that the furthest segment is Sf and the line from CoM to Sf is denoted as Lf. Relay placement works in rounds. In each round, relays are populated on the lines, starting from the line with the largest di. In the first round, a relay is first placed on Lf at a distance r away from Sf. Then, df value of Lf is updated as $df = df - r$. Because df is changed, it updates the sorted list of lines by inserting Lf into the correct index of the list. A heap data structure is used to maintain the

correct ordering of the lines. Then, the connectivity status of Sf is updated based on whether the relay that has been just placed on Lf can reach any of the neighboring partitions. If the RN cannot reach a neighboring segment (either left or right), the connectivity status of Sf will not be changed. This process will continue until the connectivity status of Sf is changed. If the relay on Lf can reach a segment Sr to the right ofSf, Sf becomes right connected and Sr becomes left connected. The direction of relay placement for both Sf and Sr will be different from this point forward. Because Lr is shorter than Lf, when the turn of Sr comes, the direction for relay placement will be tilted toward the right, because Sr is left connected. In addition, in the next round, the direction of relay placement for Sf will be tilted to the left, because it has already become right connected.

To illustrate how the algorithm works, assume that the sorted list is {$L1$, $L3$, $L5$, $L6$, $L2$, and $L4$}. In the first round, it deploys RN $R1$ along the line $L1$ and updates the list as {$L3$, $L5$, $L6$, $L2$, $L4$, $L1$}. Next, it will deploy RNs $R2$, $R3$, $R4$, and $R5$ along the lines $L3$, $L5$, $L6$, and $L2$, respectively, and update the list (see Fig. 20a). None of these relays can reach any segment. However, after deploying $R5$, it updates the status of $L1$ and $L2$, because these two lines are connected through $R1$ and $R5$ (i.e., $L1$ is right connected, and $L2$ is left connected). Therefore, updating the list is seized, and the final deployment order of the lines will be {$L4$, $L1$, $L3$, $L5$, $L6$, $L2$}. Similarly, it updates the status of $L3$ and $L4$ after $R6$ has been placed. When it is the turn of $L1$, it deploys $R7$ to its left neighbor $L6$, because $L1$ is right connected (see Fig. 20b).

After deploying $R7$, it updates the status of $L1$ and $L6$ as connected and right connected, respectively. Then, it deploys $R8$ (see Fig. 20c) and update $L2$ and $L3$. Because $L5$ is not connected so far, we deploy $R9$ along $L5$ (see Fig. 20d). Similarly, it deploys $R10$ and $R11$ toward $L5$, because $L6$ and $L4$ are right and left connected, respectively (see Fig. 20e, f). It stops when all lines are connected. Note that, unintentionally, $R11$ is also connected to $R3$, which increases the average node degree of the network. Finally, it checks if there exists a disconnected segment. If so, it connects this segment by filling the gap between that segment and closest node with additional relays. In this way the network is again connected as shown in Fig. 20f.

The execution-time complexity of the 1C-SpiderWeb heuristic is $O(s\,log\,s[d/r])$, where s is the number of segments,d is the length of longest line, and r is the relay node transmission range.

1C-SpiderWeb has following advantages:

- It uses less number of steps to federate the network.
- The execution-time complexity is very less.

1C-SpiderWeb has following drawbacks:

- It uses large number of relay nodes for federating the network.
- It does not consider terrain constraint in the network while federating network.

(b) 2C-SpiderWeb Heuristic Although ensuring 1-connectivity with robust topology feature is crucial for application-level requirements, it still does not guarantee a certain level of fault tolerance. For instance, the resulting topology may still contain cut vertices. The failure of any of these cut vertices would again partition the network and would thus disrupt the data delivery. Because this case may not be tolerated in applications such as search and rescue, backup paths are desirable. One of the solutions for this problem is thus to provide k-vertex connectivity, which can be defined as follows.

Definition: k-Vertex Connectivity: A graph is *k-vertex connected* if there are at least k vertex disjoint paths between every pair of nodes. Establishing a k-connected relay network is

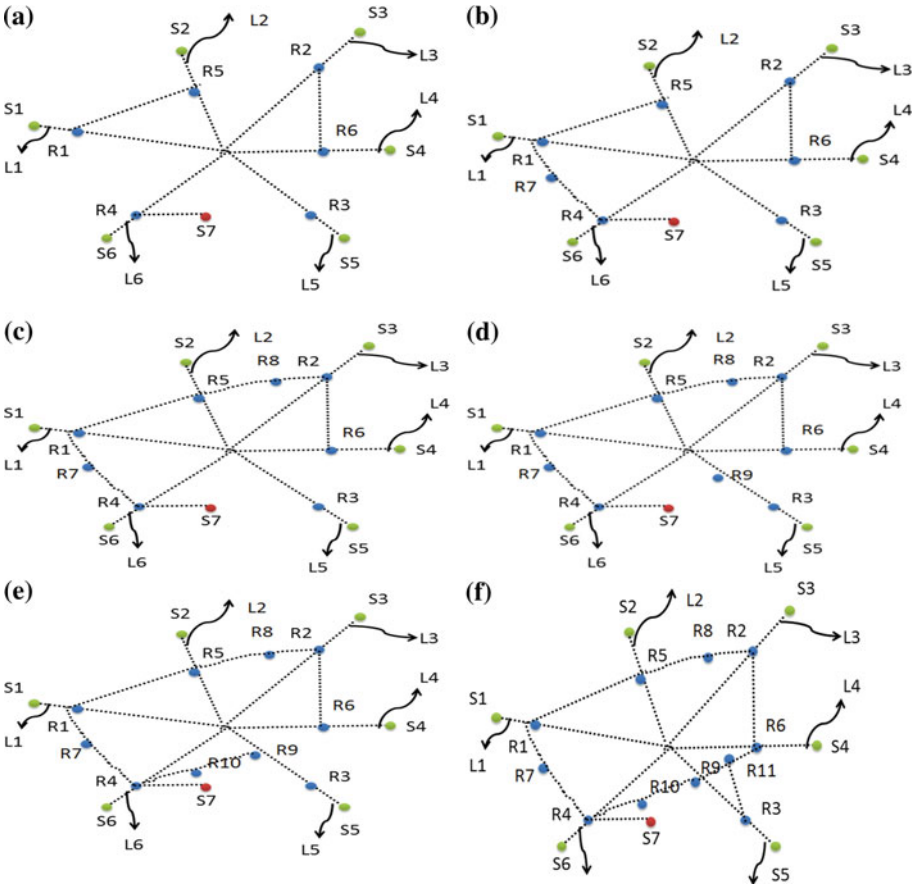

Fig. 20 **a** *Green nodes* are the segment representatives, and *Blue nodes* are RNs. Note that S7 i.e. *red node* is not involved in the algorithm, because it is an inner partition representative. Six RNs are deployed. **b** First L1 is processed, and because it is right connected, R7 is deployed toward L6. **c** Next, L3 is processed, and R8 is deployed toward L2, because L3 is right connected. **d** Because L5 is not connected; R9 is deployed toward the CoM. **e** L6 is processed, and R10 is deployed toward L5. **f** L4 is processed, R11 is deployed toward L5, and the algorithm terminates

NP hard. Therefore, authors pursue heuristics and extend 1C-SpiderWeb to provide 2-vertex connectivity. This new heuristic is referred to as 2C-SpiderWeb.

Problem Statement: Given m disjoint segments of sensors in an area of interest, determine the least count and location of RNs such that both the resulting inter-relay topology and the topology of all nodes are 2-vertex connected while maintaining some desirable features, e.g., minimized average path length, coverage, and balanced traffic load. Authors guarantee not only the 2-connectivity of the whole network (i.e., network of RNs and segments) but the 2-connectivity of the inter-relay topology as well. This property is called as *dual 2-vertex connectivity*.

Definition: Dual 2-Vertex Connectivity: Let $G = (V, E)$ be the network of partitions and RNs. G is said to be dual 2-vertex connected if and only if both G and its inter-RN network are 2-vertex connected. This property is very crucial, because a 2-connected inter-relay network will serve as a backbone among disjoint segments and can help in ensuring

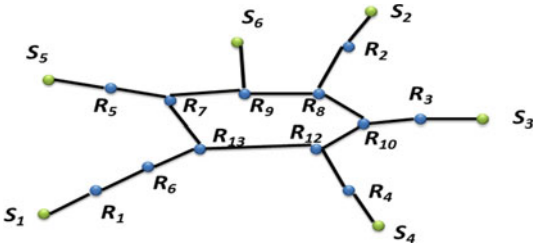

Fig. 21 Sample topology formed from 1C-SpiderWeb

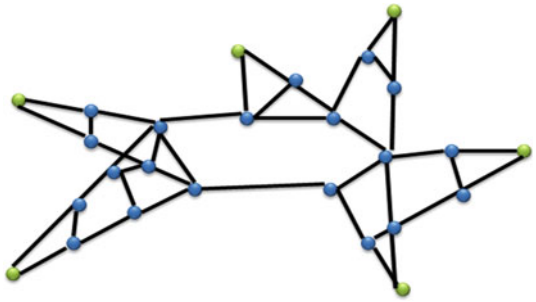

Fig. 22 Topology after running 2C-SpiderWeb i.e. each segment is connected to the second closest RN to establish an additional vertex-independent path to the ring

reliable data routing among the segments and providing a certain quality of service (QoS), e.g., real-time data delivery and reliability for the applications.

The proposed heuristic for forming a 2-vertex connected intersegment topology operates in two stages by first employing 1C-SpiderWeb to establish connectivity and then achieving 2-connectivity by carefully placing additional relays. As presented in the previous section, 1C-SpiderWeb establishes the connectivity by forming a ring of RNs around the CoM of the segments. By connecting the segments to the ring with a path, a connected network is formed, as shown in Fig. 21. From a segment point of view, this path is the only path to the ring. Let p_i be the shortest path from segment *Si* to the ring, e.g., $p_1 = R_1 \rightarrow R_6 \rightarrow R_{13}$ in Fig. 21. Because a ring is already 2-vertex connected network, the cut vertices must belong to p_i. Let C be the set of cut vertices, i.e.

$$C \subseteq Up_i : \forall sti \in n$$

where n is the number of segments.

Therefore, 2C-SpiderWeb first invokes 1C-SpiderWeb and then forms an alternative path p_i' from segment S_i to the ring such that $p_i' \cap p_i = \emptyset$. In addition, the first elements of p_i and p_i' must be connected to support dual 2-vertex connectivity, as shown in Fig. 22. We classify the vertices in G_R by labeling them with labels *first*, *cut*, and *ring*.

First: First node of every p_i'. *Cut*: All the nodes of p_i, except for the corresponding *first* and *ring-entrance* nodes. *Ring*: All nodes of the ring.

The classification is performed while running 1CSpiderWeb. As we have discussed in 1C-SpiderWeb, RNs are iteratively deployed along the line that connects a segment to the CoM until the segments are connected to their left and right neighbors. It marks all the RNs that are deployed in the first iteration with the *first* label. The nodes that are deployed in later iterations will be marked with the *cut* label. In 1C-SpiderWeb, line statuses are updated after deploying each RN. If the connectivity status of line L_i changes upon deployment of Ri, it mark Ri with *ring* label. As aforementioned, 2C-SpiderWeb forms an alternative path p_i' for each S_i. In essence, for each segment S_i, the algorithm finds the second closest *ring* node

and fills the gap between them. If the first nodes of p_i' and p_i are not connected to each other, then it will deploy additional nodes between these two nodes. At the end of each round, it label newly added nodes with the *ring* label and update the labels of the nodes of p_i with the *ring* label. Figure 22 shows the connected network after 2C-SpiderWeb heuristic.

6.2.2 Distributed Optimized Relay Node Placement Using Minimum Steiner Trees (DORMS)

Lee et al. [40] suggested first distributed approach called Distributed Optimized Relay Node Placement using Minimum Steiner Trees (DORMS) for large scale damage in the network. The objective of DORMS is to federate partitioned network while deploying movable relay nodes (RNs) from each segmented part towards center of deployment (CoD) and as soon as RNs come in the range of each other the partitioned segments connected, and resume operation. The main goal of DORMS is to design an efficient topology in terms of path length with minimum count of required RNs for federation. For minimization of deployed relay nodes, the authors have used approximation algorithm based minimum Steiner tree. Unlike centralized approached DORMS does not using flooding to search partitioned segments after damage. DORMS consists of three phases:

(i) *Initial deployment of relay node:* DORMS does not make any assumption about network, therefore each segment does not about the other segments locations after damage. DORMS initially deploys RNs along the shortest path from each segment towards center of network with a distance r, where r is range of RN. Figure 23a shows an illustration of isolated network. The shortest path between each segment (Seg_i) (to CoD is a line. RNs move from each (Seg_i) to CoD reconnect the segments. In order to coordinate the participation of RNs, a leading RN is used that defines the segment identification (SID_i) (based on segment location (Loc_i) to distinguish from other deployed RNs of different segments. The leading RN identifies itself as $RNid$, where d is an index along path P_i to distinguish the order of RNs i.e. forSeg_0, the leading RN will be RN_{io} as shown in Fig. 23b. Before moving towards CoD, RN_{io} sends a message START_MOVE to closest RN in its segment and includes its rank (i.e. value of index d). RN receiving this message from RN_{io} increased d by one and makes itself RN_{i1} and then move towards CoD. Before moving, notifies its nearest relay and repeats this process till RN reaches either on CoD before any other segment RN or RNs of different segments are not connected. The number of RNs on a path P_i) will be almost $[Length(P_i)/r] - 1$. Effectively, only one leading RN will reach to CoD and rest will establish connectivity prior to reach CoD.
Before a leading RN terminates its initial phase, it sends a STOP_MOVE message to its neighbor RNs having same SID_i).

(ii) *Optimized Deployed Relay Nodes*: After the initial phase of deploying the RNs along a path towards CoD, the second phase starts in order to reduce the number of deployed RNs. DORMS uses MST to identify essential RNs for inter-segment connectivity and returns the unnecessary RNs back to their respective segments so that these can be used in future incase of further failure. To start this phase, CoD RN waits for all leading RNs to arrive at a position r unit away from itself. The waiting time $T_{pause} = \{v\ x\ Length(max\,Dist)/2\} - \{s\ x\ Length(my\,Dist)\}$, where v is a common speed of RNs, $(max\,Dist)$ /2 and $my\,Dist$) are diagonal of deployment area and distance in which CoD has moved respectively. After waiting of T_{pause} seconds, CoD sends a message $INIT\_RN$ to all neighboring RNs. Upon receiving the message each leading node broadcasts its information stored in Loc_Seg table (i.e$\{RN_{i0}, SID_i, Loc_i\}$) .Based upon$Loc_Seg$, a leading RN identifies

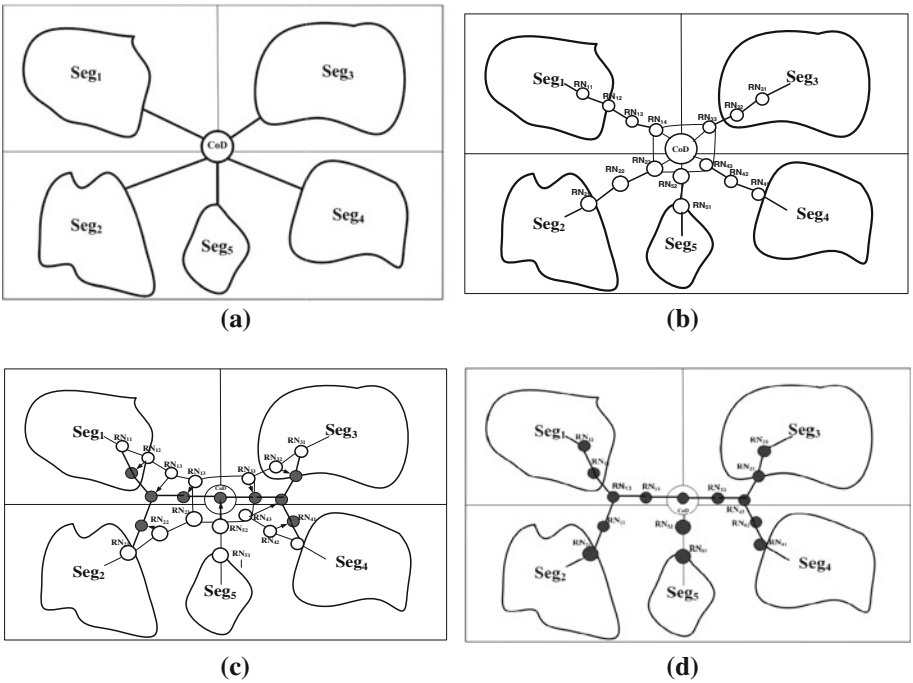

Fig. 23 **a** Segmented Networks. **b** Initial inter-segmented topology created after RNs deployed from each segment towards the CoD. **c** Relocation of deployed RNs using MST. **d** Connected network after recovery

the segment position and finds an MST for two segments and CoD. Figure23c shows how to find MST for three vertices to relocate the RNs.

(iii) *Relocation of deployed Relay Nodes*: This phase is used to reduce the overall number of relay nodes during the initialization phase. In order to reduce the number of deployed RNs without severing the resorted connectivity, DORMS uses MST that favors fewer RNs. Based on selected MST_{ij}, for $\{Seg_i, CoD, Seg_j\}$, DORMS relocated the deployed relays along P_i and P_j. In order to minimize the relocated time and re-constructed the new topology as quickly as possible, DORMS uses two leading RNs, RN_{i0} and RN_{j0} jointly find the closest RN in MST. Figure 23d shows updated topology, after relocation phase.

DORMS has following advantages:

- Due to distributed in nature, recovery is faster as compared to existing approaches.
- Message complexity is low due to distributed nature.

DORMS has following drawbacks:

- Redundant RNs are required for federation of partitioned network.
- Obstacles are not considered in the network.

6.2.3 An Optimized Based approach (Network Flow-based Model)

Sir et al. [41] proposed a solution in terms of mathematical model of non-linear programming (MINLP) which minimizes the total distance travelled by nodes for federating the partitions. The idea is based on network flow model with following objectives:

(i) Minimize the total distance travelled by nodes for connecting partitioned network (i.e. $Min \sum_{\forall i \in S} Mi$).
(ii) Minimize the maximum travel distance moved by actor node (i.e $MinMax_{i \in S} M_i$).

To understand the approach in brief, let us assume a partitioned network as shown in Fig. 23a having many partitions due to flood or fire. To recovery from these partitions, the authors have used following objective formulations:

$$Min \sum_{\forall i \in S} Mi \tag{1}$$

$$Min \quad Max_{i \in S} M_i \tag{2}$$

Subject to

$$M_i \leq r\, y_{ij} + M\left(1 - y_{ij}\right) \forall i, j \in S, i \neq j \tag{3}$$

$$\sum_{j \in S} f_{ij} = n - 1 \tag{4}$$

$$\sum_{i \in S, i \neq j} f_{ij} = 1 + \sum_{k \in S, k \neq j} f_{jk} \, \forall j \in S \tag{5}$$

$$f_{ij} \leq (n-1)\, y_{ij} \, \forall i, j \in S, i \neq j \tag{6}$$

$$y_{ij} \in \{0, 1\} \, \forall i, j \in S, i \neq j \tag{7}$$

$$f_{ij} \geq 0 \forall i, j \in S, i \neq j \tag{8}$$

The objective functions (1) and (2) are used to minimize total travelling distance and maximum distance moved by an actor node. Constraint (3) determines distance between a pair of actors and limit to range (r). y_{ij} is used to tell the connectivity between actor nodes. If y_{ij} is 1, the distance between ith actor and jth actor is less than equal to (r), otherwise 0. M is very large integer value. Constraint (4) ensures that $n - 1$ items are shipped out of the source and delivered to other actor in the network. Constraint (5) ensures the flow balance between actor nodes. Constraints (6) and (7) ensure that flow from one actor to another is possible if they are in the communication range. Constraint (8) ensures that flow variable should be positive for all actor nodes.

Flow-based approach has following drawbacks:

- Due to centralized nature of this approach network experiences large communication overhead, this causes congestion in the network.
- There is a risk of single point failure in the network.
- This approach is not scalable, and support only 30 nodes and maximum 5 partitions.

6.2.4 Optimized Relay Node Placement for Connecting Disjoint Networks (ORC)

Lee et al. [42] suggested a novel approach Optimized Relay node placement algorithm using a minimum Steiner tree on the Convex hull (ORC). ORC strives to identify Steiner points (SPs) on which relays are deployed such that the segments connect with the least number of relays. ORC deploys RNs inwards from the periphery of the area identified by the convex hull. Distributed schemes have been developed to recover a loss of a single node by pursuing the relocation of its neighbors. However, these schemes cannot handle the case when multiple nodes fail simultaneously. To understand this scheme considers a partitioned network as shown in Fig. 24.

When a set of collocated nodes get damaged by uncontrolled event/force, their 1-hop neighbors detect this failures in the network. These neighboring nodes may assess the scope

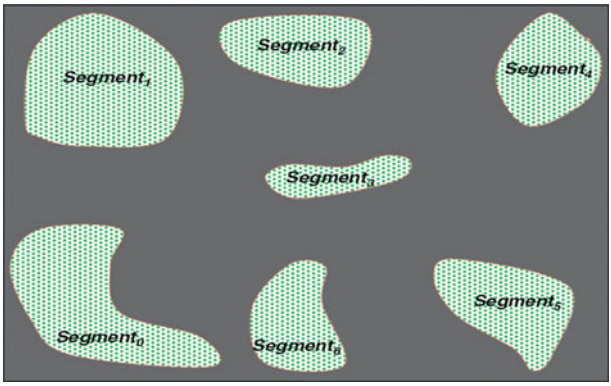

Fig. 24 Illustration of a partitioned network due to large scale damage; the dark area represents damaged area and seven segments become disconnected [42]

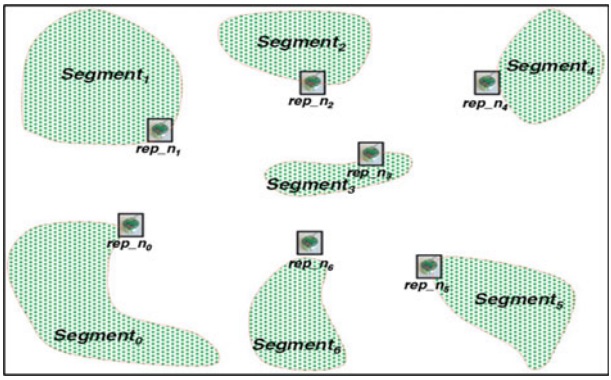

Fig. 25 A rep_n_i illustrates a representative node of the corresponding segment i [42]

of the failure by correlating consecutive node failures in the vicinity, by witnessing a major and sudden drop in communication traffic and/or by noticing un-reachability of a certain set of remote nodes. Upon confirming the multi-node failure and partitioning of the network, these neighbors become the border nodes of the damage area and broadcast a message on active links to notify all reachable nodes, which will naturally belong to their segment. After some pre-determined convergence time the border node that lost connection to the most number of neighbors becomes a representative node. This can be tracked by including the number of neighbors in the notification messages that get flooded in the segment (i.e. local flooding). The rationale of the representative node selection is that new RNs are deployed in the vicinity of these border nodes and it is thus imperative to restore the network topology as similar as possible to its pre-failure. If a tie occurs, the border node which has more live neighbors and/or the highest Node_Id becomes a representative node. To determine the location of network partitions, it is assumed that the network has a few mobile nodes, e.g., robots that will search the area on behalf of the individual segments. The navigation of these mobile nodes can used GPS, if they are equipped with a receiver, or on a relative coordinate system developed using localization techniques.

ORC pursues greedy heuristics and has two main phases: (1) Identify SPs on which RNs would be placed with the objective to minimize the number of deployed RNs to connect seg-

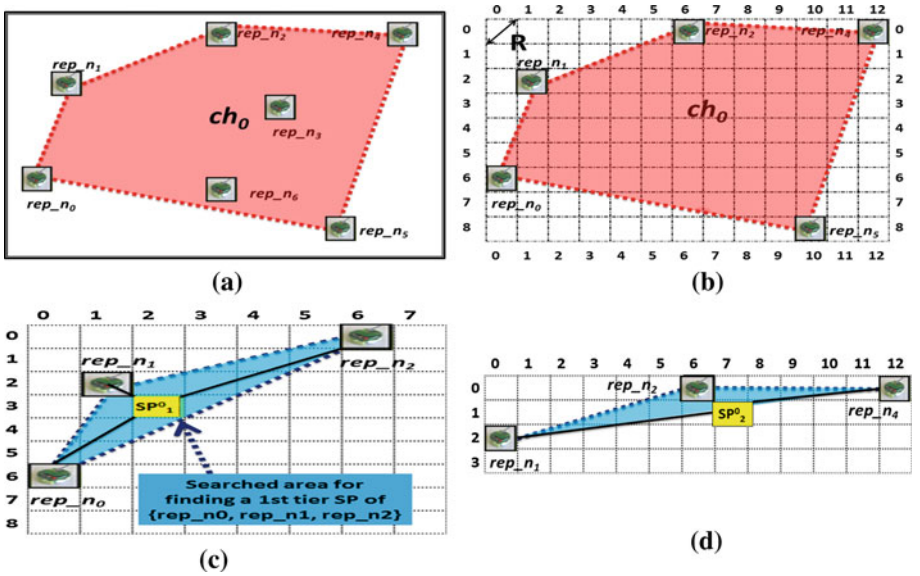

Fig. 26 **a** The rep_n$_0$, rep_n$_1$, rep_n$_2$, rep_n$_4$, and rep_n$_5$ form the convex hull Ch_0 and ORC strives to connect those representative nodes on Ch_0 using MST in the first round. **b** Grid structure of the damage area, in which each cell is square-shaped with a side length of $\frac{R}{\sqrt{2}}$, a SP will be searched among the cells. **c** 1st tier SPs for connecting {rep_n$_0$, rep_n$_1$, rep_n$_2$}. **d** 1st tier SP for connecting {rep_n$_1$, rep_n$_2$, rep_n$_4$}. **e** 1st tier SP for connecting {rep_n$_2$, rep_n$_4$, rep_n$_5$}. **f** 1st tier SP for connecting {rep_n$_4$, rep_n$_5$, rep_n$_0$}. **g** 1st tier SP for connecting {rep_n$_5$, rep_n$_0$, rep_n$_1$}. **h** Each of the solid cells located at (3, 2), (1, 7) (2, 8), (6, 8) and (5, 2) represents the 1st tier SPs for connecting rep_n$_0$, rep_n$_1$, rep_n$_2$}, {rep_n$_1$, rep_n$_2$, rep_n$_4$}, {rep_n$_2$, rep_n$_4$, rep_n$_5$}, {rep_n$_4$, rep_n$_5$, rep_n$_0$} and {rep_n$_5$, rep_n$_0$, rep_n$_1$} respectively. **i** The convex hull Ch_1 found in the second round; three points such as SP_2^0, SP_3^0, and rep_n$_3$ are connected and thus merged into one point. **j** SP_2^0, SP_3^0, and rep_n$_3$, are merged into one terminal and three SP_j^1's, SP_1^1, $SP_2^1$and SP_3^1 are found in the 2nd round (t = 1). **k** SP_2^1, and $SP_3^1$are merged into one terminal, there is no more SPs found by ORC in the 3rd round (t = 2). **l** The restored topology by ORC. The total 23 relays are finally populated to connect seven segments

ments. (2) Deploy additional RNs in order to form a connected inter-SP topology considering the communication range of a RN.

The first phase has two main steps that are repeated to determine all necessary SPs. In the first step, ORC finds the convex hull to identify the boundary segments. Then, SPs that connect every three neighboring boundary segments are identified. These SPs are referred as 1st tier SPs. The unengaged segments along with convex-hull are again computed to identify boundary terminals (segments or 1st tier SPs) and used in the second round to find 2nd tier SPs. The 3rd tier SPs will be identified based on the 2nd tier and so on. In other words, the two steps are repeated recursively for t rounds until the number of points considered for computing a convex hull is less than three or they form a complete graph in terms of a communication range of a RN. ORC then switches to the second phase in which the identified SPs and segments are stitched together. Basically, every segment $Segi$ identifies the closest SP and RNs get placed on the line from $Segi$ to such SP. The same process applies for the 1st tier SP to connect them to the 2nd tier and so on. For example in Fig. 25, the representative nodes are rep_n$_0$, rep_n$_1$, rep_n$_2$, rep_n$_3$, rep_n$_4$, rep_n$_5$ and rep_n$_6$ for different segments.

ORC operates in rounds. In the first round (t=0), ORC identifies a set of segments in the damaged area, which forms the smallest polygon that contains the other segments. Consid-

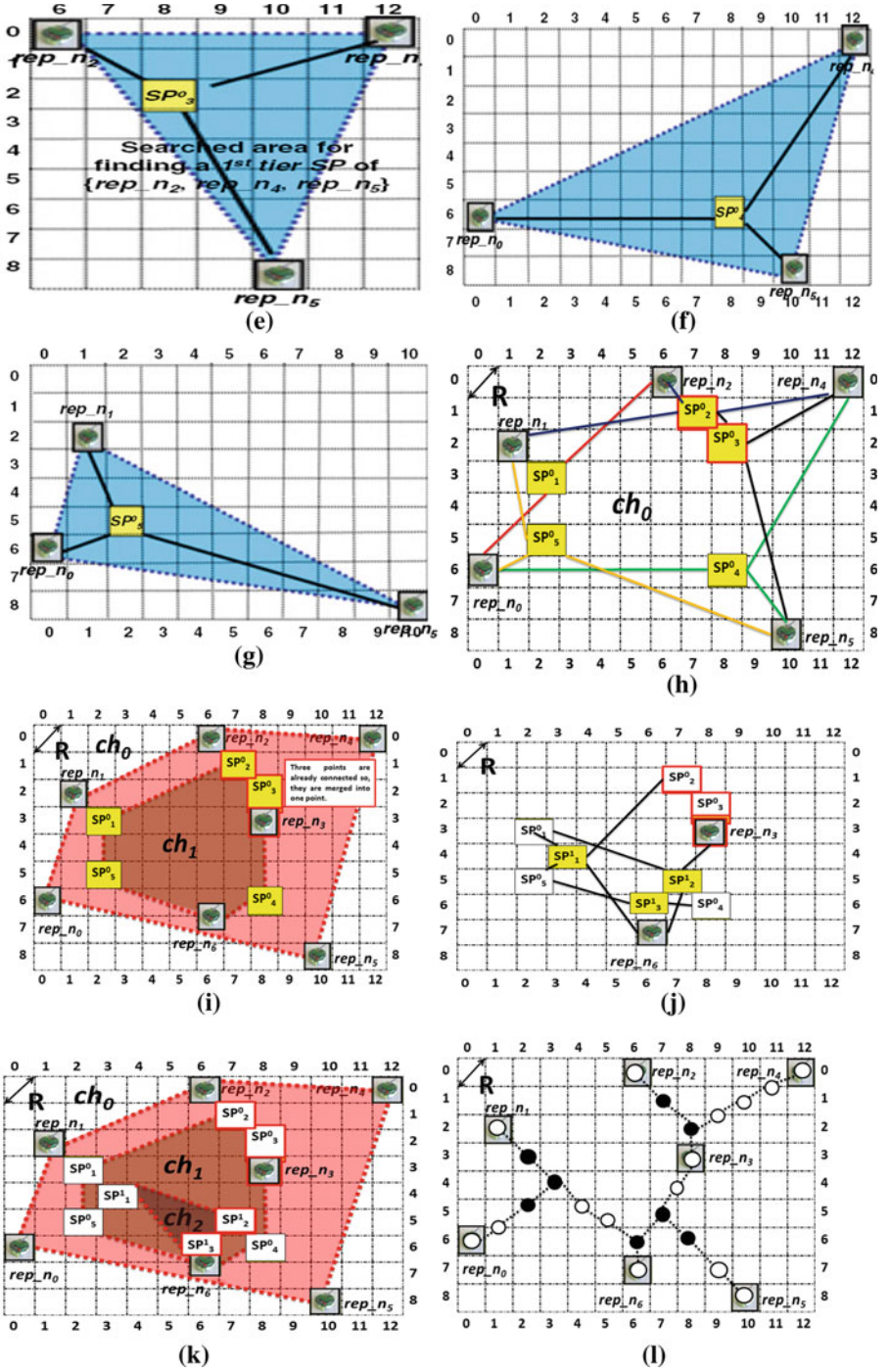

Fig. 26 continued

ering the segments as terminals, the convex hull of all segments is used to identify boundary segments. It is assumed that there exist at least three non-collinear segments, so that the convex hull Ch_0 found in the first round forms a closed polygon as seen in Fig. 26a. To find a convex hull, a well known algorithm Graham scan algorithm is used in this approach.

ORC then strives to connect every set of three adjacent border segments using the k-restricted loss-contracting algorithm (k-LCA), which is the best known approximation algorithm to solve a Steiner tree problem (STP). The rationale for considering three terminals at a time is the k-LCA algorithm, which is used to find SPs, yields better performance when the number of terminals (k) is small and its execution time grows exponentially with the increase in k. Therefore, to identify the SPs that connects the boundary segments onCh_0, ORC opts to find a minimum spanning tree (MST) for each 3-tuple, {rep_n_0, rep_n_1, rep_n_2}, {rep_n_1, rep_n_2, rep_n_4}, {rep_n_2, rep_n_4, rep_n_5}, {rep_n_4, rep_n_5, rep_n_0}, and {rep_n_5, rep_n_0, rep_n_1} as shown in Fig. 26b. The found SPs will be referred to as the 1st tier SPs. In order to find the MST that connects three terminals using k-LCA, ORC quantizes the polygon defined by the convex hull. Basically, ORC identifies the smallest rectangle L that includes the polygon and modeled it as a grid. A cell in the grid is square-shaped with a side length of $R/\sqrt{2}$. The grid cells become then a set of candidate SPs out of which the 1st tier SPs are identified by the k-LCA algorithm. The motivation is that the least number of RNs would be placed along the lines between the found SP and each of the three terminals and thus the distance between the centers of two diagonally neighboring cells is set to R.

Based on graph (G), ORC then strives to find the 1st tier SPs among the vertices in V that minimize the number of RNs required for connecting rep_n_x, rep_n_y, and rep_n_z, where each of x, y and z is an index of three neighboring boundary segments on Ch_0.

Figure26c explains how MST is found for the three segments rep_n_0, rep_n_1, and rep_n_2. ORC first identifies the cells which are completely or partially covered by the triangle (rep_n_0, rep_n_1, rep_n_2), namely {(0, 6), (1, 3), (1, 4), (1, 5), (2, 1), (2, 2),(2, 3), (2, 4), (3, 1), (3, 2), (3, 3), (4, 0), (4, 1), (4, 2), (5, 0),(5,1), (6,0)}. k-LCA then tries each Steiner tree which contains any cell "c" in T as an SP. For seeking the Steiner tree, a minimum spanning tree over {rep_n_0, rep_n_1, rep_n_2, c} is computed. The resulting tree has rep_n_0, rep_n_1, and rep_n_2 as leaves with c connected to each of those terminals. The k-LCA algorithm concludes that cell (3, 2) connects the three representative nodes with the least cost. Namely, the three nodes and the found 1st tier SP, SP_1^0 form an MST as seen in Fig. 26c.

Similarly, other SPs are found by k-LCA to connect each tuple of three rep_n_i's, {rep_n_1, rep_n_2, rep_n_4}, {rep_n_2, rep_n_4, rep_n_5}, {rep_n_4, rep_n_5, rep_n_0} and {rep_n_5, rep_n_0, rep_n_1} respectively. The shaded triangle and the solid lines in Fig. 26d–g shows the searched area to find the 1st tier $SP_i^0$and the MST formed by the SPs respectively.

Therefore, SP_1^0, SP_2^0, SP_3^0, SP_4^0, and SP_5^0 are found in the cells (3, 2), (1, 7), (2, 8), (6, 8), and (5, 2) respectively in the first round as seen in Fig. 26h.

In the second round (t = 1), the unengaged segments, namely, rep_n_3 and rep_n_6, and the five SP_i^0's; $I = 1 \ldots 5$ found in the first round are considered and ORC computes the convex hull $Ch_1$for them. Before finding SPs, ORC investigates the connectivity between terminals on the convex hull. For instance, SP_2^0, SP_3^0, and rep_n_3 in Fig. 26i represents neighboring cells in the grid and thus relays placed in $SP_2^0$and SP_3^0, will be able to reach each other and reach the rep_n_3. Therefore, these cells are merged into one terminal. Thus, ORC opts to find five MSTs as shown in Fig. 26i.

In the second phase, ORC forms a connected topology using the least RN count. Basically, the distance between the SPs identified in the first phase and between them and the representative nodes of the various segments may exceed the communication range of the RNs. Therefore, ORC opts to federate RNs along the shortest paths which connect each segment,

rep_n_i to the nearest SP, SP_i^1. Since the SP^1i's are the points through which rep_n_i would be connected to its 1 or 2 hop neighboring border segments with the least cost, the boundary segments would be connected to each other with the least RN count via the second phase. This process is further applied for connecting the SPs and segments in the individual rounds. In other words, the least number of RNs are populated along the shortest paths which connect the 1st tier SPs to the nearest 2nd tier SPs, connect the 2nd tier SPs to the nearest 3rd tier SPs and so on. Figure 26k shows the resulting inter-segment topology which is a minimum spanning tree of seven representative nodes, rep_n_i's, i = 0 . . . 6 and eight SP_i^t's t = 0, 1. In Fig. 26l, the black circles represent RNs which are deployed on SPs and the relays placed in the white ones connect segments and SPs. ORC finally places a total of 23 RNs to connect seven segments.

6.2.5 Game-Theoretic Based Approach

Senturk et al. [43] have proposed a novel centralized approach to restore connectivity between various partitions based on game theory heuristic. The idea is based on comparison of Nash-equilibrium (i.e. equilibrium value of a partition) of various partitions and the partition having largest equilibrium will be stationary while nodes that are in less equilibrium are relocated for federating the partitioned network. In this way, partition nodes itself can be used to connect the partitions until network is connected. For calculating partition equilibrium two parameters are considered (1) Node degree: It is the number of 1-hop neighbors of a node. (2) Central Point (CP): It is defined as the average number of nodes' x and y coordinates. The distance (i.e. CP) along with nodes' degree is used to compute the equilibrium of a node. Initially, all nodes may have different equilibrium values depend on their degrees and locations in the network. However, despite having different equilibrium values, all nodes in the same partition have same goal i.e. restoration of connectivity. Therefore, a single equilibrium value can be used as Nash-equilibrium for each partition. This can be obtained by adding all equilibrium values within that partition as shown below:

$$\mathbf{e}_{\boldsymbol{Par}_\mathbf{j}} = \sum_{\mathrm{i} \in |\mathrm{Par_j}|} \deg_\mathrm{i} * \mathrm{dis_i} - \mathrm{CP}$$

where $|Par_j| \in n$ and n is number of nodes in jth partition, $1 < j > N$ and N is total number of nodes in the network, CP central point as explained earlier.

When equilibrium of all partitions are calculated, nodes in the partition having second largest value of equilibrium relocate towards the partition of largest equilibrium until connectivity restored. Afterwards, a new equilibrium is computed for federating next partition. This process continues until all partitions are connected. Total moving distance is used to calculate moving cost of nodes.

6.2.6 Biological-Inspired Optimization for Sensor Lifetime (BiO4SeL)

Castro et al. [44] proposed first distributed network partitioning recovery approach with routing protocol to handle multi-failure nodes in WSNs. The approach is based on swam intelligence a very well known bio-inspired heuristic i.e. Ant Colony Optimization (ACO). BiO4SeL is different from other algorithms in the sense that it integrates routing protocol with recovery protocol to tackle nodes' failure problem in the network (i.e. cut-vertex nodes). It uses battery power information of each node for routing table updates as well as for partition recovery. In other words, it uses node stability for packet forwarding and

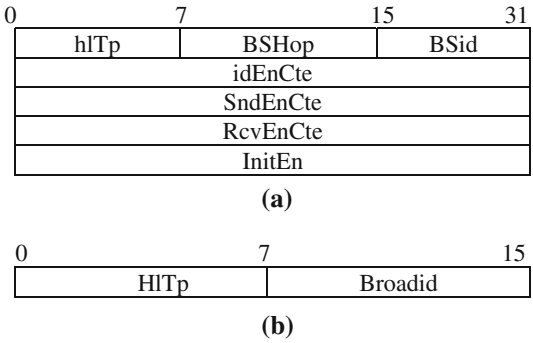

Fig. 27 **a** Packet Format of hello packet. **b** Packet Format of iant packet

route discovery as well as recovery in the network. The objective of BiO4SeL is to build and maintain a probabilistic routing table which is used by nodes to distribute packet relaying responsibility among several nodes. When a node makes its choice for next hop, it uses probability value maintained on each node instead of always choosing the next shortest distance node (i.e. using in shortest path). With this way, the algorithm is able to distribute overhead among nodes in the network. BiO4SeL has mainly three phases as follows:

(i) *Initialization phase:* In this phase, the nodes start its operation with broadcast an initial hello message to all its neighbors i.e. when Time-To-live (TTL) equals 1. The hello message consists of packet type (hlTp) and its own battery information (InitEn) as shown in Fig. 27a, b which may change from one sensor to another sensor. idEnCte (idle), SndEnCte (sending) and RcvEnCte (receiving) are model specific consumption rate of sensor nodes. After this phase, each node aware of its reachable neighbors along with battery information.

(ii) *Route discovery phase:* In this phase each node will discover its reachability to destination using pheromone value (i.e. battery data).

(iii) *Data forwarding phase:* Before this phase, initial route and routing tables are already computed, and are just waiting for data sent. At each unicast, the table is queried and a node is probabilistically chosen based on path pheromones. As packet is forwarded, the energies and pheromones are updated and node will wait for next operation. In this way packet will be forwarded and routes are chosen if any failure occurs in the network.

BiO4SeL has following drawbacks:

- It uses mobile agents to find the stability of node i.e. stability of route, which generates more communication overhead in the network.
- Each node has to maintain more information about the network i.e. space complexity is more.
- BiO4SeL assumes multiple routes are available in the network which is sometime not available in WSANs due to sparse nature of actor nodes in the network.

7 Comparison Summary of Various Approaches

Many proposals have been presented in the existing scenario that provide some good level of recovery mechanisms with some constraint, but further more researches need to be done

towards developing a system that can be used without or less constraint. Moreover, It is observed from the survey study that no approach is optimal that satisfy all performance metrics for recovery purpose. Although, some approaches are good in terms one or more metric(s), but poor for another metric(s) and even no scalable. Table 2 and 3 shows the comparison tables of different approaches (i.e. single nodes' failure and multi-node failure) based on their performance parameters. The following section gives the detail description of these metrics used in different literatures for evaluation.

- *Approach Type:* This parameter illustrates the type of technique i.e. centralized or distributed used in the network for recovery operation. The localized distributed approaches are better for recovery in WSANs due to low communication overhead (store local information about the network i.e. 1-hop or 2-hop) and consequently prolong the network lifetime. On the other hand, centralized approaches are better for small scale networks due to fast recovery (i.e. every node has complete knowledge of network). Furthermore, distributed approaches are proactive, reactive and hybrid types depend upon the behavior of node(s) during recovery in WSANs.

From the literature survey of single node's failure approaches, it has been observed that maximum approaches are either reactive or proactive and fewer are hybrid as shown in Fig. 28a, b. Hybrid techniques are better suit for delay-tolerant real-time applications. Therefore, more hybrid approaches are required to propose and develop with low message complexity for future applications. Moreover, in multi-node failure approaches, more distributed approaches are required to develop for network efficiency as observed in Fig. 28c.

- *Total Moving distance by Actor Nodes (TMI):* This metric demonstrates the total distance travelled by involved nodes until the connectivity is restored. The total moving distance should not change drastically even with higher node density (i.e. as number of nodes are increased). The proactive approaches (like DARA, PADRA, and PADRA+, PADRA-DP, NORAS) shows large total moving distance and even also increases with the network size due to more number of nodes have to move in the recovery which decreases network lifetime. On the other hand, in multi-node failure approaches, less number of RNs with small distance movement should be used to federate network partitioning problem. It has been observed from this study that fewer approaches are available which follow this criterion. Therefore, more approaches have to be proposed and developed in the future with different heuristics for network efficiency.
- *Number of Actors moved (NAM):* This parameter depicts the number of nodes that are involved in recovery plan. Number of moving nodes is the function of energy consumption in the network i.e. network lifetime. More energy will be consumed if large nodes move in the network. CCRA, LeMoToR and LDMR show small number of actor nodes for recovery with large communication overhead (due to flooding) in the network. DCR has less travelling nodes with small communication overhead due to fact that every node maintains 1-hop neighbors' information for network recovery. In the multi-node failure approaches, DORMS shows small number of actor nodes for recovery process with high time complexity (due to optimal calculation of MST of each segment so that relocation of RNs may further be done and decreases the number of RNs for restoring the network partitioning). Therefore, further optimization of number of moving nodes is required to enhance network efficiency and lifetime.
- *Percentage of Coverage Reduction:* Coverage is equally important like connectivity. This metric shows the reduction of coverage relative to pre-failure level as function of number of nodes travelling for recovery. More the number of travelling node(s) for recovery, larger

will be the reduction in coverage as compared to pre-failure level. Only C^2AM has considered coverage with connectivity restoration of WSANs. The main motivation behind all multi-node failure approaches except BiO4SeL, 1C-SpiderWeb and 2C-SpiderWeb to restore connectivity without considered coverage hole in the network. Therefore, more distributed coverage-aware connectivity restoration approaches are required to be proposed to integrate both coverage and connectivity.

Scalability: Scalability in the network means supporting the large number of nodes without influence on the network performance. If the network is not scalable, the following problems can happens in the network:

- *Congestion and load balancing problem:* Every node has certain data processing and storage capability i.e. amount of packets it can receive, process and forward during certain period of time. If node receives so much data from its neighboring nodes and cannot able to forward that much data in the network, then congestion may happen and packet loss may start with time. This may happens due to non-uniform distribution of nodes in the network; some part of the network is heavily dense and some part is sparse. This problem is called load balancing problem.
- *Increased routing path length:* Sometimes, if the deployment area of the network does not increase and only density of the nodes increases while adding the extra nodes to the network, it increases the path length.

Table 2 illustrate that LeMoToR, C^2AM, VCR, DCR and LDMR are more scalable than other existing approaches. Table 3 demonstrates that game-theoretic based approach and BiO4SeL are more scalable in the existing multi-node failure approaches with high message complexity. However, few approaches are well for medium size of network (i.e. up to 100 nodes) i.e. less scalable. Therefore, some new scalable approaches are required to propose and develop to take the full advantage of WSANs.

- *Message Complexity:* It represents the total number of messages that are exchanged among the nodes during recovery process. This metric gives the communication overhead during recovery and more precisely energy dissipation of the whole network. The lifetime of the network also depends on how much communication occurs in the network. It has been already proved that communication consumes more energy that computation.
  It has been observed that maximum approaches show large communication complexity during their operation in the network which is very crucial for resource constrained network. However, some single node's failure approaches show less message complexity due to distributed nature of these protocols. As shown in Table 2, the message complexity of distributed approaches (DARA, PADRA, AOM, RIM, C^2AM, VCR, CCRA, LeDiR, LDMR, LeMoToR) is $O(n)$, where n is number of nodes in the network, whereas time complexity of centralized approaches (MCDS, RSTP) is $O(n^2)$(refer Appendix A for proof) which increases exponentially with the network size. Therefore, new approaches are needed to be developed with low message overhead in the network.
- *Reaction Time*: This metric demonstrates about the recovery time of the protocols (i.e. when failure handler will react for recovery and how). Recovery time depends upon reaction time of node for recovery. Fast reaction time starts the early recovery process which create unnecessary computing in the network and slow reaction time starts recovery late which cause unnecessary recovery delay in the network. Therefore, this parameter must be taken into account and optimal value of reaction time is important for good recovery approaches without affect QoS network performance parameters.

Table 2 Single node and two non-overlapping Nodes Failure Recovery Approaches with Performance Metrics

S. No	Approach Name and Year of Publication	Approach type	Selection of failed node	Selection of closest dominatee approach used	Cut vertex determination	Message complexity	Time complexity	Total distance travelled (TMI)
1	Minimum Connected Dominating Set Approach (MCDS) (2007)	Centralized	–	Flooding	Flooding	High $O(n^2)$	High $O(r/s +(p +t)n^2)$	High
2	Distributed Actor Recovery Approach (DARA) (2007)	Distributed and localized proactive	Greedy	CDS	Local flooding	Low O(n)	Low O(r/s +(p +t)n)	High
3	(a) Partition Detection and Recovery Algorithm (PADRA) (2008)	Distributed and localized Proactive	Greedy	Greedy	Local Flooding	Low O(n)	Low O(r/s +(p +t)n)	High
	(b) Partition Detection and Recovery Algorithm (PADRA+) (2008)		Greedy	Depth First Search	Flooding	High $O(n^2)$	High $O(r/s +(p +t)n^2)$	High
	(c) Partition Detection and Recovery Algorithm-Dynamic Programming (PADRA-DP)(2008)		Greedy	Dynamic programming	Local flooding	Low O(n)	Low O(r/s +(p +t)n)	High
	(d) Multiple Partition Detection and Recovery Algorithm (MPADRA)(2008)		Mutual exclusion	Mutual exclusion	Local flooding	–	Low O(r/s +(p +t)n)	High
4	AOM (2009)	Distributed hybrid	Greedy	CDS	Local	Low O(n)	Low O(d/(r/2s) +(p +t)n)	Low

Table 2 continued

S. No	Approach Name and Year of Publication	Approach type	Selection of failed node	Selection of closest dominatee approach used	Cut vertex determination	Message complexity	Time complexity	Total distance travelled (TMI)
5	Recovery through Inward Movement (RIM) (2008)	Self-healing localized and distributed hybrid	Greedy	Greedy	Local	Low O(n)	Low O(r/v + α)	High $\frac{r}{2}\sum_{i=1}^{n} hi$
6	Connectivity with Application level Constraints on Actor Mobility (C^2AM) (2009)	Self-healing, lightweight and localized distributed reactive	Greedy	Predicate rules based	Local	Low O(n)	Low O(r/s +(p +t)n)	Low
7	Volunteer Instigated Connectivity Restoration algorithm (VCR) (2010)	Distributed and localized reactive	Greedy	Greedy	Local	Low O(n)	Low O(r/s +(p +t)n)	High
8	Least Disruptive Topology Repair Algorithm(LeDiR) (2010)	Distributed and localized reactive	Greedy	Greedy	Local flooding	Low O(n)	Low O(r/s +(p +t)n)	High
9	Coordinated-Assisted Connectivity Recovery Approach (CCRA) (2011)	Distributed and localized reactive	Greedy	Greedy	Local	Low O(n)	Low O(r/s +(p +t)n)	Low
10	Least Movement Topology Repair Algorithm (LeMoToR) (2011)	Distributed and localized reactive	Greedy	Greedy	Local	Low O(n)	Low O(r/s +(p +t)n)	Low

Table 2 continued

S. No	Approach Name and Year of Publication	Approach type	Selection of failed node	Selection of closest dominatee approach used	Cut vertex determination	Message complexity	Time complexity	Total distance travelled (TMI)
11	Least Distance Movement Recovery Algorithm (LDMR) (2011)	Distributed and localized reactive	Greedy	Greedy	Local flooding	Low O(n)	Low O(r/s +(p +t)n)	$\left(\frac{r}{2} * n\right) + \sum_i^n hij$
12	Coverage Conscious Connectivity Restoration Algorithm (C^3R) (2010)	Coverage conscious distributed and localized reactive	Greedy	Spare nodes through greedy approach	Local	Low O(n)	–	High
13	Node Recovery through Active Spare designation (NORAS) (2010)	Distributed proactive	Greedy	Spare Nodes through DFS	Flooding	Low O(n)	–	High
14	Rapid Spanning Tree Protocol for Link failure recovery (RSTP) (2011)	Centralized	–	Tie-set	Flooding	Medium O(n+ m)	–	–
15	Distributed Connectivity Restoration Algorithm (DCRA) (2011)	Distributed reactive	Mutual exclusion	Collision avoidance	Local	–	–	Low
16	(a) Distributed Partition Detection and Connectivity Restoration Algorithm (DCR)	Distributed localized hybrid	Greedy	Greedy	Local	Low O(n)	Low O(nr)	Low
	(b) Recovery algorithm for multiple node failures (RAM) (2012)		Mutual exclusion	Node criticality and availability (NCA)	Local flooding	High $O(n^2)$	High $O(n^2r)$	High

Table 2 continued

S. No	Approach Name and Year of Publication	TMI complexity	Individual distance travelled (MMI)	Node degree	False alarm	Reaction time	Mean time to failure (MTTF)	Scalability
1	Minimum Connected Dominating Set Approach(MCDS) (2007)	Low O(nr)	High	Yes	No	No	No	No
2	Distributed Actor Recovery Approach (DARA) (2007)	Low O(nr)	Low	Yes	No	No	No	No
3	(a) Partition Detection and Recovery Algorithm (PADRA) (2008)	Low O(nr)	Low	Yes	No	No	No	No
	(b) Partition Detection and Recovery Algorithm (PADRA+) (2008)	High $O(n^2r)$	High	Yes	No	No	No	No
	(c) Partition Detection and Recovery Algorithm-Dynamic Programming (PADRA-DP) (2008)	Low O(nr)	High	Yes	No	No	No	No
	(d) Multiple Partition Detection and Recovery Algorithm (MPADRA) (2008)	High $O(n^2r + m)$	High	Yes	No	No	No	No
4	AOM (2009)	Low O(nr/2)	Low	Yes	Yes	No	No	Less scalable

Table 2 continued

S. No	Approach Name and Year of Publication	TMI Complexity	Individual distance travelled (MMI)	Node Degree	False Alarm	Reaction Time	Mean time to failure (MTTF)	Scalability
5	Recovery through Inward Movement (RIM) (2008)	Low O(nr)	High	Yes	No	Low	Yes O(H.r)	No
6	Connectivity with Application level Constraints on Actor Mobility (C^2AM) (2009)	–	Low	No	No	No	No	No
7	Volunteer Instigated Connectivity Restoration algorithm (VCR) (2010)	Low O(nr)	High	Yes	No	No	No	Less scalable
8	Least Disruptive Topology Repair Algorithm(LeDiR) (2010)	Low O(nr)	High	Yes	No	No	No	No
9	Coordinated-Assisted Connectivity Recovery Approach (CCRA) (2011)	Low O(nr)	Low	Yes	No	No	No	Less scalable
10	Least Movement Topology Repair Algorithm(LeMoToR) (2011)	Low O(nr)	Low	Yes	No	No	No	High

Table 2 continued

S. No	Approach Name and Year of Publication	TMI complexity	Individual distance travelled (MMI)	Node Degree	False Alarm	Reaction time	Mean time to failure (MTTF)	Scalability
11	Least Distance Movement Recovery Algorithm (LDMR) (2011)	Low O(nr)	Low	Yes	No	No	No	High
12	Coverage Conscious Connectivity Restoration Algorithm (C^3R) (2010)	Low O(nr)	High	No	No	No	No	Less Scalable
13	Node Recovery through Active Spare designation (NORAS) (2010)	Low O(nr)	High	Yes	No	No	No	No
14	Rapid Spanning Tree Protocol for Link failure recovery (RSTP) (2011)	Low	–	–	–	–	–	No
15	Distributed Connectivity Restoration Algorithm (DCRA) (2011)	High	Low	–	–	–	–	High
16	(a) Distributed Partition Detection and Connectivity Restoration Algorithm (DCR)	Low O(nr)	Low	Yes	No	No	No	Yes
	(b) Recovery Algorithm for Multiple node Failures (RAM) (2012)	High $O(n^2r)$	Low	Yes	Yes	No	No	No

Where r range of an actor node, n number of actor nodes, r/v round trip time, α Constant time for synchronization, H network diameter, hi number of hops from each neighbor to the leaf node (network diameter) assuming all neighbors have disjoint paths to leaf nodes, dij distance between the corresponding non cut-vertex node and the position of the nominating neighbor node, m number of edges in the network, s is number of segments after partition, '−' means not defined, $N \in \{1, 2, 3 \ldots M\}$ number of partitions in the network

Table 3 Multiple Nodes Failure Recovery Approaches with Performance Metrics

S. No	Approach Name and Year of Publication	Approach type	Selection of closest dominatee approach used	Cut vertex determine	Message complexity	Time complexity	TMI
1	Bio-inspired based approaches(1C-spider, 2C-spider) (2011)	Centralized	Spider based Heuristic	Convex hull	High	Low	Low
2	Distributed Optimized Relay Node Placement Using Minimum Steiner Trees(DORMS) (2010)	Distributed reactive	Minimum Steiner Tree Approach	Heuristic Based on MST	Low	High	Low
3	An Optimized Based approach(Flow-based Model) (2011)	Centralized	Flow-based network model	Transportation model	High	High	High
4	Optimized Relay Node Placement for connecting disjoint networks(ORC) (2012)	Centralized	Minimum Steiner tree on convex hull	Greedy heuristic	Low	Medium	Low
5	Game-Theoretic based approach (2012)	Centralized	Game Theory	Nash-Equilibrium	High	Low	High
6	Biological–Inspired Optimization for sensor Lifetime(BiO4SeL) (2012)	Distributed reactive	Distributed ant colony optimization (DACO)	Flooding	High	Low	–

Table 3 continued

S. No	Approach Name and Year of Publication	Throughput and end-to-end delay	Number of partitions considered	Nodes complexity for federating partitioning	Reaction time	Mean time to failure (MTTF)	Scalability
1	Bio-inspired based approaches(1C-spider, 2C-spider) (2011)	No	Large	High	No	No	No
2	Distributed Optimized Relay Node Placement Using Minimum Steiner Trees(DORMS) (2010)	No	Large	Medium (i.e. 20–30)	No	No	High
3	An Optimized Based approach(Flow-based Model) (2011)	No	Small Up to 5 Partitions	High	Yes	No	No
4	Optimized Relay Node Placement for connecting disjoint networks(ORC) (2012)	No	Large	Medium	No	No	Less scalable
5	Game-Theoretic based approach (2012)	No	–	Less	No	No	Less scalable
6	Biological–Inspired Optimization for sensor Lifetime(BiO4SeL) (2012)	Yes	–	No	Yes	Yes	High

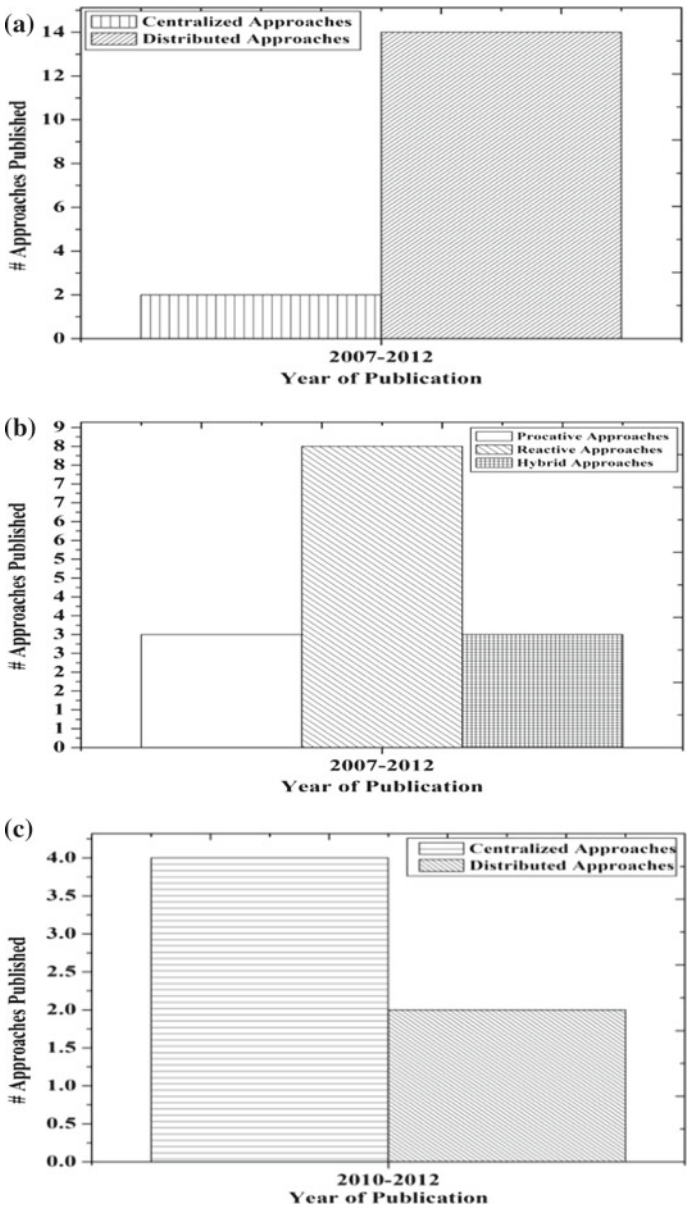

Fig. 28 **a** Single node failure approaches versus year of publication. **b** Single node failure proactive, reactive and hybrid approaches versus year of publication. **c** Multi-node failures approaches versus year of publication

Very few approaches (i.e. RIM) have considered reaction time for recovery in single node failure approaches. Moreover, in multi-node failure recovery approaches, only BiO4SeL has included reaction time indirectly for nodes' failure recovery. Therefore, more approaches with optimal reaction time have to be proposed for real time applications.

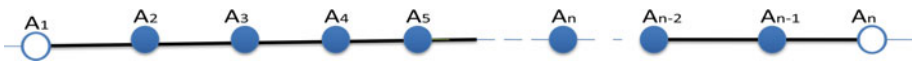

Fig. 29 Worst case network topology

8 Conclusion and Future Work

Network partitioning recovery is a key technology and hot topic of WSANs in recent years. Given the importance to network partitioning recovery problem in WSANs of critical applications and also the lack of any classifications in the existing literatures, we have proposed a comprehensive survey work by focusing on detailed description of network partitioning recovery approaches. The categorization has been derived from existing research studies in the area of connectivity, coverage, coordination and communications in WSANs. Moreover, this categorization shows that proper network partition recovery is feasible without much challenged conditions. The lessons learned from this surveyed work is that mutual cooperation and controlled mobility along with connectivity of actor nodes must be taken into account more seriously in order to beneficially utilize the full potentials of WSANs in many real life applications. In the future, we will try to propose new optimized approach based on present research gaps and try to evaluate in WSANs simulation environment based on actual network parameters such as throughput, delivery ratio, end-to-end delay and latency etc.

Acknowledgments This work is supported by the Department of Computer Engineering, National Institute of Technology, Kurukshetra, Haryana (India) and Department of Computer Science and Engineering, Thapar University, Patiala, Punjab (India). We would like to thanks all the authors of the research papers and reviewers who advised us directly or indirectly to write this detailed survey paper in a fruitful manner.

9 Appendix

Theorem 1 *The worst case of total moving distance of all the actor nodes involved in recovery process for DARA, PADRA, PADRA+, PADRA-DP, C²AM, RIM, CCRA, VCR, LeDiR, LeMoToR, LDMR and optimal solution is* $O(nr)$.

Proof The worst case topology for all approaches is, when all the nodes are in a line as shown in Fig. 29.

In this topology, assuming that fails, then in the worst case failure handler (FH) would be $A5$ which triggers the movements till is hit. Thus TMI is given by $\sum_{i=1}^{n-4} r = (n-4)r = O(nr)$, assuming that the distance between the nodes is equal to maximum of transmission range (r).

For optimal solution $A3$ will be picked as failure handler (FH). Thus TMI is given by $3r$ which is again $O(nr)$. The worst case of optimal solution would be the case where the failed node is in the middle of line topology. In this case TMI is given by $\sum_{i=1}^{n+1/2-1} r = \left(\frac{n-1}{2}\right) r$ which is $O(nr)$. In AOM approach, the travelling distance of a node is $r/2$. Therefore, TMI is given by

$$\sum_{i=1}^{n-4} \frac{r}{2} = (n-4)\frac{r}{2} = O(n\frac{r}{2}).$$

□

Theorem 2 *The worst case time complexity of DARA, PADRA, PADRA+, PADRA-DP, C^2AM, RIM, CCRA, VCR, LeDiR, LeMoToR, LDMR is $O(\frac{r}{s} + (p + t)n)$ where s is speed of nodes, p is the propagation delay for distance r and t is the transmission delay.*

Proof In Fig. 29, let us assume that failure of a node $A4$ has been occurred then Failure handler (FH) will send a message to its closest neighbors gets an acknowledgment, and move to the failed node. Similarly the node receiving this message will do the same thing and move to its parent position which had been moved to failed node. Therefore, replacement can be done parallel. Therefore, when the node gets the message, it will take $\frac{r}{s}$ time to reach to its parent position, where r is range of the node and is speed of the node. Now to calculate the time to reach the message to the node will be $3(p + t)$ as each node is sent three messages. Since, in above topology there can be at most $(n - 4)$hops, and then total time will be $O(\frac{r}{s} + 3(n - 4)(p + t))$which is$O(\frac{r}{s} + (p + t)n)$. □

Theorem 3 *The worst case message complexity of DARA, PADRA, PADRA-DP, C^2AM, RIM, CCRA, VCR, LeDiR, LeMoToR, LDMR is $O(n)$ where n is number of nodes in the network.*

Proof Consider worst case topology as shown in Fig. 29. Let us assume that $A2$ is failed and $A3$ act as failure handler. Then, $A4$ will start a replacement process until An is not hit. This means $(n-2)$ nodes will be replaced. At each replacement a request and an acknowledgement are used. Thus $2(n - 2)$messages will be sent to restore the connectivity in worst case. Adding 4n messages for CDS and n messages for closest node designation, then total message complexity will be $2(n - 2) + 4n + n = 7n - 4$ which is $O(n)$. □

Theorem 4 *The worst case message complexity of MCDS, PADRA+ is $O(n^2)$ where n is number of nodes in the network.*

Proof In MCDS and PADRA+, a DFS is performed for each cut-vertex from $A3$ to $An - 1$ again with respect to Fig. 29. Thus number of nodes performing DFS will be$(n - 3)$. For A_2, this will cost traversing each node until An is hit. The total number of messages for $An - 2$ is $(n - 1)$ as well as we assume that $An - 3$will start the DFS. As a result, only two nodes will introduce a total of $(n - 1)$ messages. Therefore, total cost will be $(n - 1) + (n - 2) + \cdots + (n - (n - 3)/2) + \cdots + (n - 2) + (n - 1)$ Which is $2|\sum_{i=1}^{\frac{(n-3)}{2}} (n - i)| = O(n^2)$ □

Theorem 5 *RIM would successfully restore connectivity, if $RTmax(recoverytime) < E[(P(t)]$where $E[(P(t)]$is the expected value of the probability distribution function $P(t)$ of the node failure.*

Proof RIM is guaranteed to successfully yield a connected network topology, if the time between nodes' failure is longer than $RTmax$. Using mean time between successive failures $(MTBF)$ as a node reliability measure, a connectivity restoration condition for RIM can be derived as follow:

$$RTmax \leq MTBF$$

where MTBF can be calculated using probability density function $P(t)$ of node failure.

$$MTBF = E[P(t)]$$

Therefore, $RTmax < E[P(t)]$. □

References

1. Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., Cayirci, E. (2002). A Survey on Sensor Networks. *IEEE Communications (Magazine), 40*(8), 102–116 .
2. Kay, R., & Mattern, F. (2004). The design space of wireless sensor networks. *IEEE Wireless Communications, 11*(6), 54–61.
3. Akyildiz, I. F., & Kasimoglu, I. (2004). Wireless sensor and actuator networks: Research challenges. *Ad-hoc Networks*, *2*(4), 351–367.
4. Petriu, E. M., Georganas, N. D., Petriu, D. C., Makrakis, D., & Groza, V. Z. (2000). Sensor-based information appliances. *IEEE Instrumentation and Measurement Magazine*, *3*(4), 31–35.
5. Díaz, M., Garrido, D., Llopi, L., Rubio, B., & Troya, J. M. (2006). A component framework for wireless sensor and actuator network. In *Proceedings of 11th IEEE international conference on emerging technologies and factory automation* (ICETFA'06).
6. Yuan, H., Huadong, M., & Hongyu, L. (2006). Coordination mechanism in wireless sensor and actuator networks. In *Proceedings of first international multi-symposiums on computer and computational sciences* (IMSCCS'06).
7. Ruiz-Ibarra, E. & Villasenor-Gonzalez, L. (2008). Wireless Sensor and Actor Networks II. In *Proceeding of IFIP international federation for information processing*, vol. 264 (pp. 62–73). Boston: Ali Miri.
8. Karthickraja, N. P. & Sumathy, V. (2010). A study of routing protocols and a hybrid routing protocol based on rapid spanning tree and cluster head routing in wireless sensor networks. In *Proceedings of ICWCSC'10*.
9. Salarian, H., Chin, K.-W., & Naghdy, F. (2012). Coordination in wireless sensor-actuator networks: A survey. *Journal of Parallel and Distributed Computing, 72*(7), 856–867.
10. Jie, W., Ming, G., & Stojmenovic, I. (2001). On calculating power-aware connected dominating sets for efficient routing in ad hoc wireless networks. In *Proceedings of IEEE international conference on parallel processing* (pp. 346–354).
11. Li, L., & Halpern, J. Y. (2001). Minimum-energy mobile wireless networks revisited. In *Proceeding of IEEE international conference on, communications (ICC'2001)* (pp. 278–283).
12. Ya, X., Heidemann, J., & Estrin, D. (2001). Geography-informed energy conservation for ad hoc routing. In *Proceeding of annual international conference on mobile computing and networking, (MOBICOM'01)* (pp. 70–84).
13. Yuanyuan, Z., Jia, X., & Yanxiang, H., (2006). Energy efficient distributed connected dominating sets construction in wireless sensor networks. In *Proceedings of international conference on wireless communications and mobile computing*, ACM.
14. Chen, B., Jamieson, K., Balakrishnan, H., & Morris, R. (2002). Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *Wireless Networks*, *8*(5), 481–494.
15. Ong, J., You, Y. Z., Mills-Beale, J., Tan, E. L., Pereles, B., & Ghee, K. (2008). A wireless, passive embedded sensor for real-time monitoring of water content in civil engineering materials. *IEEE Sensors Journal*, *8*, 2053–2058.
16. Lee, D. S., Lee, Y. D., Chung,, W.-Y., & Myllyla, R. (2006). Vital sign monitoring system with life emergency event detection using wireless sensor network. In *Proceedings of 5th IEEE Conference on Sensors*, Daegu, Korea, 22–25 October.
17. Hao, J., Brady, J., Guenther, B., Burchett, J., Shankar, M., & Feller, S. (2006). Human tracking with wireless distributed pyroelectric sensor. *IEEE Sensors Journal*, *6*, 1683–1696.
18. Stankovic, J. A. (2008). When sensor and actuator networks cover the world. *Invited paper in the proceeding of, ETRI Journal, 30*(5), October.
19. Chen, J., Diaz, M., Llops, L., Rubio, B., & Troya, J. M. (2011). A survey on quality of service support in wireless sensor and actor networks: Requirements and challenges in the context of critical infrastructure protection. *Journal of Network and Computer Applications (Elsevier)*, *34*, 1225–1239.
20. Younis, M., & Akkaya, K. (2008). Strategies and techniques for node placement in wireless sensor networks: A survey. *Ad-Hoc Networks*, *6*(4), 621–655.
21. Senel, F., Akkaya, K., & Younis, M. (2007). An efficient mechanism for establishing connectivity in wireless sensor and actor networks. In *Proceedings of IEEE GLOBECOM'07* (pp. 1129–1133).
22. Abbasi, A. A. Akkaya, K., & Younis, M. (2007). A distributed connectivity restoration algorithm in WSANs. In *Proceeding of 32nd IEEE conference on local, computer networks* (LCN'07) (pp. 496–502).
23. Akkaya, K., Thimmapuram, A., Senel, F., & Uludag, S. (2008). Distributed recovery of actor failures in WSANs. In *Proceeding of IEEE WCNC'08* (pp. 2480–2485).
24. Akkaya, K., Senel, F., Thimmapuram, A., & Uludag, S. (2010). Distributed Recovery from Network Partitioning in Movable Sensor/Actor Networks via Controlled Mobility. *IEEE Transactions on Computers, 59*(2), 258–271.

25. Zamanifar, A., Kashefi, O., & Sharifi, M. (2009). AOM: An efficient approach to restore Actor-to-Actor connectivity in WSANs. *International Journal of Computer Networks and Communications (IJCNC), 1*(1), 61–72.
26. Jorgic, M., Stojmenovic, I., Hauspie, M., & Simplot-ryl, D. (2004). 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, workshop* (pp. 360–371).
27. Younis, M., Lee, S., Gupta, S., & Fisher, K. (2008). A localized self-healing algorithm for networks of moveable sensor nodes. In *Proceeding of IEEE GLOBECOM'08*.
28. Abbasi, A. A., Baroudi, U., Akkaya, K., & Younis, M. (2009). C^2AM: An algorithm for application-aware movement-assisted recovery in wireless sensor and actor networks. In *Proceedings of ACM, "IWCMC'09"*, June 21–24, Leipzig, Germany.
29. Imran, M., Younis, M., Said, A. Md., & Hasbulla, H. (2010). Volunteer-instigated connectivity restoration algorithm for wireless sensor and actor networks. In *Proceedings of 8th IEEE international conference on embedded and ubiquitous, computing* (EUC'10) (pp. 679–683).
30. Abbasi, A., Younis, M., & Baroudi, U. (2010). Restoring Connectivity in Wireless Sensor-Actor Networks with Minimal Topology Changes. In *Proceedings of IEEE international conference on communications (ICC'10)* (pp. 1–5). South Africa: Cape Town.
31. Zhao, X., & Wang, N. (2011). Coordination-assisted connectivity recovery approach in wireless sensor and actor networks. In *Proceeding of 3rd international conference on computer research and development (ICCRD'11)*, vol. 4 (pp. 82–86). Shangai, China.
32. Abbasi, A. A., Younis, M., & Baroudi, U. (2011). Restoring connectivity in wireless sensor-actor networks with minimal node movement. In *Proceedings of 7th international wireless communications and mobile computing conference (IWCMC'11)* (pp. 2046–2051). Turkey: Istanbul.
33. Alfadhly, A., Baroudi, U., & Younis, M. (2011). Least distance movement recovery approach for large scale wireless sensor and actor networks. In *Proceedings of 7th international wireless communications and mobile computing conference (IWCMC'11)* (pp. 2058–2063). Turkey: Istanbul.
34. Tamboli, N., & Younis, M. (2010). Coverage-aware connectivity restoration in mobile sensor network. *Journal of Network and Computer Applications (Elsevier)*, *33*, 363–374.
35. Vaidya, K, & Younis, M. (2010). Efficient failure recovery in WSNs through spare designation. In *Proceedings of 1st international workshop on interconnections of wireless sensor networks* (IWSN'10), Santa Barbara, CA, USA, June 2010.
36. Kadena, K., & Nakayama, K. (2011). *Proceeding of workshops of international conference on advanced information networking and applications* (pp. 467-472), Fukuoka, Japan.
37. Mi, Z., Yang, Y., & Guangjun, L. (2011). HERO: A hybrid connectivity restoration framework for mobile multi-agent networks. In *Proceedings of IEEE international conference on robotics and automation* (pp. 1702–1707). Shanghai, China.
38. Imran, M., Younis, M., Said, A. Md, & Hasbullah, H. (2012). Localized motion-based connectivity restoration algorithm for WSANs. *Journal of Network and Computer Applications*, *35*, 844–856.
39. Senel, F., Younis, Mohamed, & Akkaya, Kemal. (2011). Bio-inspired relay node placement heuristics for repairing damaged wireless sensor networks. *IEEE Transactions on Vehicular Technology*, *60*(4), 1835–1848.
40. Lee, S., & Younis, M. (2010). Recovery from multiple simultaneous failures in wireless sensor networks using minimum Steiner tree. *Journal of Parallel and Distributed Computing (Elsevier)*, *70*(5), 525–536.
41. Sir, M. Y., Senturk, I. F., Sisikoglu, E., & Akkaya, K. (2011). An optimization-based approach for connecting partitioned mobile sensor/actuator networks. In *Proceeding of 3rd IEEE international workshop on wireless sensor, actuator networks and robot networks (WiSARN'11)* (pp. 525–530). Shangai, China.
42. Lee, S., & Younis, M. (2012). Optimized relay node placement for connecting disjoint wireless sensor networks. *Computer Networks*, *56*, 2278–2804.
43. Senturk, I. F., Yilmaz, S., & Akkaya, K. (2012). A game-theoretic approach to connectivity restoration in WSANs. In *Proceedings of IEEE international conference on communications (ICC'12)* (pp. 7110–7114). Ottawa, ON, Canada.
44. de Castro, M. F., Riberio, L. B., & Oliveira, C. H. S. (2012). An autonomic bio-inspired algorithm for WSN self-organization and efficiency. *Journal of Network and Computer Applications*, *35*, 2003–2015.

Author Biographies

Virender Ranga received his B.Tech degree in Electronics and Communication from Kurukshetra University Kurukshetra in 2001 and M.Tech in Computer Science and Engineering from Guru Jambeshwar Technical University Hissar in 2004 respectively. Currently, he is Assistant Professor in Computer Engineering Department NIT Kurukshetra Haryana, India. He is working towards his PhD in Sensor Adhoc Network from the Department of Computer Engineering, National Institute of Technology Kurukshetra, India. He has published more than 10 research papers in various International/National journals and conferences. He has guided more than 10 M. Tech. dissertations and currently guiding four M.Tech dissertations inside as well as outside the institute. His interest areas are distributed recovery in Sensor Networks, Fault Tolerance in Sensor Networks, Coverage and Connectivity in Sensor and Actor Networks and QoS in WSANs.

Mayank Dave obtained his M.Tech. degree in Computer Science and Technology from IIT Roorkee, INDIA in 1991 and Ph.D. from the same institute in 2002. He is presently working as Associate Professor in Department of Computer Engineering at NIT Kurukshetra, INDIA with more than 19 years experience of academic and administrative affairs in the institute. He is presently Associate Professor in Computer Engineering Department. He has published approximately 85 research papers in various International/National journals and conferences. He has coordinated several projects and training programs for students and faculty. He has delivered number of expert lectures and keynote addresses on different topics. He has guided four PhDs and several M.Tech. dissertations. His research interests include Peer-to-Peer Computing, Pervasive Computing, Wireless Sensor Networks and Database Systems.

Anil Kumar Verma is currently a faculty in the department of Computer Science and Engineering at Thapar University, Patiala. He received his B.S., M.S. and Doctorate in 1991, 2001 and 2008, respectively, majoring in Computer Science and Engineering. He has worked as Lecturer at M.M.M. Engg. College, Gorakhpur from 1991 to 1996. He joined Thapar University in 1996 and is presently associated with the same Institute. He has been a visiting faculty to many institutions. He has published over 100 papers in referred journals and conferences (India and Abroad). He has chaired various sessions in the International and National Conferences. He is active member of MIEEE, MACM, MISCI, LMCSI, MIETE, GMAIMA. He is a certified software quality auditor by MoCIT, Govt. of India. His research interests include wireless networks, routing algorithms and mobile clouds.