Yan *et al. EURASIP Journal on Wireless Communications and Networking* (2017) 2017:138
DOI 10.1186/s13638-017-0921-4

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Hybrid connectivity restoration in wireless sensor and actor networks

Ke Yan, Guangchun Luo*, Ling Tian, Qi Jia and Chengzong Peng

Abstract

Wireless sensor and actor networks are becoming more and more popular in the recent years. Each WSAN consists of numerous sensors and a few actors working collaboratively to carry out specific tasks. Unfortunately, actors are prone to failure due to harsh deployment environments and constrained power, which may break network connectivity resulting in disjoint components. Thus, maintaining the connectivity among actors is especially important. This paper proposes hybrid connectivity restoration (HCR), which integrates proactive selection and reactive motion. An actor protectively selects a backup node through its one-hop neighbor table and informs the backup node to supervise its stage. Once it fails, the backup node moves to the best position to restore the connectivity of the failed node's neighbors reactively. This triggers a local recovery process at the backup node, which is repeated until network connectivity is restored. In order to minimize travel distance, HCR selects the backup node which moves the shortest distance to restore connectivity. Furthermore, HCR opts to reduce the number of messages by just informing the failure to its backup node. The correctness and effectiveness of HCR are validated through both theoretical analysis and simulations.

Keywords: Hybrid connectivity restoration, Wireless senor and actor networks, Single-node failure

1 Introduction

Wireless sensor networks (WSNs) are indispensable components of Internet of Things [5–7, 9–11, 15, 19, 23–25, 32, 37, 38]. A wireless sensor and actor network (WSAN) is a special kind of WSN, which has motivated lots of research works [28]. In the corresponding applications such as environmental monitoring, battlefield surveillance, border protection, target searching and tracking, a number of sensors and actors work cooperatively to monitor a specific area and track a target of interest. Sensors are responsible for collecting data, and actors are responsible for processing data and bridging the sensors and the control center. An actor and the sensors connected to it form a self-organized sub-network. All the sub-networks collaborate with each other to carry out tasks. It is desired that all the actors in a WSAN are connected at any time.

Unfortunately, due to harsh deployed environments and limited battery power, actors may deplete energy fast. A sudden loss of a node may break network connectivity resulting in disjoint network components. Therefore, it is important to detect node failures and restore network connectivity as early as possible. Since WSANs are usually deployed far away from the control center and are operated autonomously and unattended, it is difficult and inefficient to control the restoring process in a centralized manner. Connectivity restoration therefore should be a distributed, localized, and self-healing process. In addition, a rapid connectivity restoration is desired in order to reduce the baneful influence of node failures. Moreover, the overhead such as the total travel distance and the total number of messages should be minimized considering the limited energy supply. The average travel distance should be considered as well because one node traveling too far will consume too much energy and may cause another network disconnection. A node failure disrupting network connectivity is called a cut vertex, which is difficult to identify in large-scale WSANs centralized and timely. Though there have been many distributed cut-vertex detection algorithms, they are time-consuming and resource-intensive. As a result, it is very challenging to restore network connectivity in a distributed, localized, and efficient manner.

*Correspondence: gcluo.uestc@gmail.com
School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China

This paper proposes hybrid connectivity restoration (HCR) considering single-node failures, which integrates *proactive selection* and *reactive motion*. The selection of a backup node for a failure node is a proactive process. Each node identifies a backup node and is then monitored by its backup node. Once a node fails, its backup node moves to the best position that connects all the failed node's one-hop neighbors. It triggers the restoration of the backup node and the restoration is a reactive process. HCR is a distributed and localized scheme, where each node just maintains its one-hop neighborhood information. Since a failed node only affects its direct neighbors' connectivity, the main idea of the restoration is to move one of the failed node's neighbors to a new position so that all the failed node's direct neighbors can be re-connected. The node motion may trigger another disconnection on the moving node, so the restoration is a recursive process, the whole network is connected only when the motion node's directed neighbors are connected. HCR opts to efficiently restore network connectivity through selecting the most proper backup node and moving it to the best position instead of the failed node's position. The less distance the node moves, the less influence on the network connection. As aforementioned, only if a node failure breaks its directed neighborhood connectivity, it may further break network connectivity. The node whose failure breaks its directed neighborhood connectivity is called a critical node. On the opposite, the node is called an uncritical node. It should be noted that the uncritical node cannot be a cut vertex while the critical node may be a cut vertex. Though a critical node's failure may not necessarily break network connectivity, it may bring unnecessary restoration. It is much more efficient and cheaper to identify a critical node and restore its direct neighbors' connectivity than to identify a cut vertex and restore the network connectivity. To identify a cut vertex requires global information, which is impossible and inefficient in WSANs. At the same time, to identify a critical node just needs one-hop neighborhood information, and the identification is done on the node itself. Moreover, compared with moving a backup node to the failed node's position in DCR [17], it is better to move the backup node to the position which connects all the failed node's directed neighbors. Only when all the neighbors are on the boundary of the communication range, the backup node needs to move to the failed node's position.

Following are our main contributions:

- We proposed a hybrid connectivity restoration in WSAN, which integrates *proactive selection* and *reactive motion*. The proactive selection of a backup node can shorten a recovery process, the HCR offers effectiveness and timeliness.
- Different from moving a backup node to the position of failed node, HCR moves the backup node to the best position with the shortest travel distance to reconnect the failed node's one-hop neighbors. This can not only reduce a motion cost but also the total overhead, as the shorter a node travels, the fewer nodes are influenced.
- HCR opts to reduce the number of messages by just informing the failure to its backup node.
- The efficiency of HCR is mathematically analyzed and validated through simulations. The bounds on the incurred overhead are derived. HCR outperforms RIM and DCR in terms of the number of relocated nodes, total travel distance, average travel distance, and number of messages for both dense and sparse networks.

The remainder of the paper is organized as follows. Section 2 describes the system model and formulates the problem. Section 3 reviews the related works. Section 4 provides the detailed description of the proposed HCR, and the theoretical analysis is illustrated in Section 5. The simulation results are presented in Section 6. Finally, the conclusions and future work are presented in Section 7.

2 System model and problem formulation

2.1 System model

A WSAN consists of numerous sensors and a few actors (Fig. 1) which are usually randomly deployed in harsh environments far away from the control center. So a WSAN is a self-organized network where sensors are mainly responsible for collecting data and sending data to the nearest actors, and actors are responsible for making decisions. Actors process data, communicate with other actors, and send the results to the sink node. In this paper, it is assumed that actors are movable to connect other actors forming a connected network. In WSANs, actors play the role of gateways to the sink node. Therefore, it is important to maintain actor connectivity. Since sensors are static, they always try to communicate with actors in their communication range without identifying who they are. That means all the actors are the same to sensors. Therefore, actor connectivity decides network connectivity. Here, the coverage of the WSAN is not considered since an actor's failure will lose all the sensors connected to it. Because the sensors are randomly deployed in the sensing regions, it can be assumed that there is a equal number of sensors within an actor's transmission range. Therefore, moving actors has a little effect on the total coverage of the WSAN.

In WSANs, each actor has limited communication radius R_c. Actors can send and receive messages within the communication range to discover other actors.

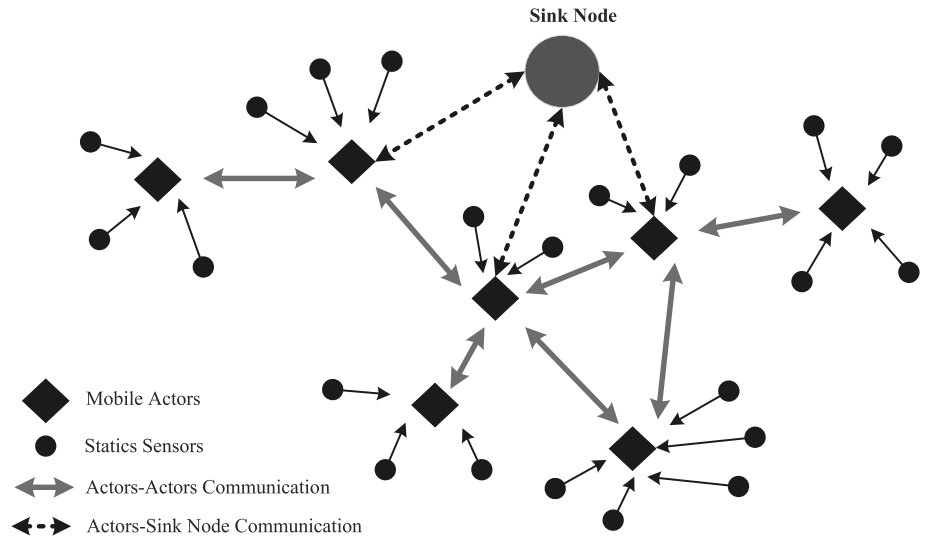

Fig. 1 A WSAN with mobile actors

Each actor maintains a one-hop neighborhood table recording its neighbors' positions and other information. Two-hop neighborhood information or multi-hop neighborhood information can be obtained by exchanging one-hop neighborhood table with neighbors. The more information it gets, the more communication and storage it incurs. However, in most WSAN applications, actors are battery powered and have limited energy. Thus, it is more efficient to maintain less information. In this paper, each actor i just maintains a one-hop neighborhood table denoted as $NT(i)$. $NT(i)$ is a two-dimensional table where each row contains one-hop neighbor information such as unique node ID (*ID*), local position (*POSITION*), and critical character (*CRITICAL*). The critical character is defined as follows.

Critical character: It indicates whether a node failure breaks neighbor connectivity. $CRITICAL = 1$ when the n ode failure breaks connectivity. Otherwise, $CRITICAL = 0$.

In order to restore connectivity rapidly, each actor also has backup node information. The selection of a backup node is introduced in Section 4.1. The one-hop neighborhood table and backup node information of each actor are maintained and updated during the process of connectivity restoration.

2.2 Problem formulation

Actors are prone to failures due to tough environments or energy depletion. The loss of an actor affects not only the sensors connected to it but also the neighboring actors. The latter case is even worse, so this paper focuses on the latter case. It is assumed that actors are movable, it is possible to restore network connectivity by relocating actors.

Generally, different node failures have different effects on network connectivity. Consider the network shown in Fig. 2. If node $A3$ fails, the network is divided into two disjoint components. If node $A8$ fails, the network is still connected and all its neighbors can communicate with each other. A leaf node's failure does not break connectivity. It is important to note that a node's failure breaking one-hop neighborhood connectivity does not necessarily mean it breaks network connectivity. For example, node $A6$ has two neighbors $A4$ and $A7$. If $A6$ fails, $A4$ and $A7$ cannot communicate with each other through $A6$, but $A5$ still connects $A4$ and $A7$. Though only a cut vertex breaks network connectivity, it needs the global network connectivity information to tell whether a node is a cut vertex which is nontrivial and inefficient in large-scale WSANs. It also incurs significant overhead in terms of computation, communication, and storage. As aforementioned, an uncritical node does not break network connectivity, and the identification of an uncritical node just needs one-hop neighborhood information. Moreover, the distance between any two sibling nodes in a WSAN is no more than $2Rc$, because both of them are in the communication

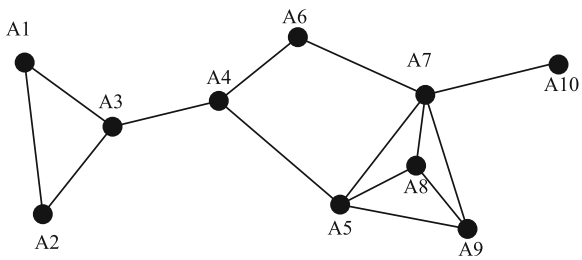

Fig. 2 A connected network of mobile actors

range of the parent node. It takes a node to move at most *Rc* towards another to reconnect it. Therefore, compared with identifying a cut vertex, it is more efficient and significant to identify a critical node.

Though reconnecting two sibling nodes just needs to move one node for less than *Rc*, it may trigger another failure, which causes cascading failures. Even if the motion does not trigger other node failures, it may consume a lot of energy. Hence, load balancing should also be taken into consideration. In addition, a node motion may generate some other critical nodes, which increases the risk of node failure in the future. Considering all of these factors, we formulate the following problem.

Given a connected WSAN $G = (V, R_c)$. *V is the set of actors, and* R_c *is the actors's communication radius. Each actor* V_i *has ID and Positions. All the actors are homogeneous with the same* R_c. *Each node just knows its one-hop neighborhood information. When a single actor i fails, relocate the rest nodes so that (1) network connectivity is restored, (2) the total travel distance is minimized, (3) the average travel distance is minimized, and (4) the total number of messages is minimized.*

HCR is proposed in Section 4 to solve this problem efficiently. It is assumed that no two or more actors fail simultaneously, and no node fails during restoration.

3 Related works

Connectivity restoration considering single-node failures has attracted much attention, and there are a lot of surveys [22, 27, 34, 36] focusing on connectivity restoration. The existing schemes can be classified into two categories: proactive restoration and reactive restoration.

Proactive restoration schemes make use of redundant resources including nodes and paths to increase the robustness of a WSAN. When a node fails, it requires no connectivity restoration because there are redundant resources maintaining connectivity. Since the directional connection between a pair of actors is determined by the communication radius, it can only result in more redundant relay nodes to build K-disjoint paths [30]. In this case, there are K-disjoint paths between any pair of actors in a WSAN. Even K-1 paths fail, there is still a path connecting them. Consider a two-connected WSAN where there are at least two paths between any pair of nodes. CRAFT [18] establishes a bi-connected inter-partition topology while minimizing the longest path length and the number of deployed relay nodes. It strives to form the largest inner simple cycle or Backbone Polygon (BP) around the center of the damaged area where no partition lies inside, and deploys relay nodes to connect each outer partition to the BP through two non-overlapping paths. The advantage of proactive restoration schemes is that it does not disturb a network when node fails, but it requires many redundant resources. The stronger fault tolerance is, the more resources are required. Moreover, it is very difficult to place relay nodes optically, as it needs the global network information. The time complexity is very high in large-scale networks. In [20], it is proven that just listing a set of feasible sites for the relays is already at least APX-hard. Though many heuristic algorithms have been proposed, such as Genetic Algorithm [12, 21], Artificial Bee Colony Algorithm [14], and Concentric Fermat Points [31], it is still extraordinarily time-consuming.

Different from the proactive restoration schemes, the reactive restoration schemes are passive and a recovery process is triggered when a node failure is detected. They do not require reserving redundant resources. The basic idea is to reconnect the failed node's neighbors. There are two kinds of approaches: cooperative communication and relocating nodes. Cooperative communication is first proposed in [8]. It allows a node to send message beyond its communication radius with the help of its neighbors. Two nodes are able to communicate if and only if the received average signal-to-noise ratio (SNR) is no lees than the fixed threshold. Signal strength diminishes with the increase of transmission distance and overlays at the destination. CSFR [33] adopts cooperative communication to restore connectivity. Taking advantage of neighbors to transport data does not increase the neighbor's energy consumption a lot. Though it has low overhead from the current perspective, it is still a long-term process which costs a lot of energy in the long run. In addition, it is unacceptable and very time-consuming to select the help nodes.

Currently, most reactive restoration schemes reconnect a network by replacing a failure node with a proper backup node through movement which is a recursive process that may relocate the rest of the nodes. Therefore, which node moves and where to move is nontrivial. Ramezani proposed a distribute method to restore connectivity by using a centralized genetic algorithm [26] at the basic station. It strives to minimize the number of mobile nodes and the average length of all nodes' paths. It is a heuristic algorithm. As mentioned before, only a cut vertex may break network connectivity. Many approaches decide whether a node is a cut vertex firstly and deal with cut vertex failure only, such as DARA [1], PDARA [2], PCR [16], and NNN [13]. DARA identifies a cut vertex through two-hop neighborhood information. Once a failure happens, the failed node's neighbors select the most proper backup node considering node degree and distance and inform its sibling nodes. In fact, the process of identifying a cut vertex is not introduced in details in DARA. In a latter improved approach PDARA, it forms a connected dominating set (CDS). PDARA informs a particular node in advance whether a partition occurs in case of failure. They both strive to localize the scope of the recovery process and minimize

the movement overhead imposed on the involved actors. In nearest non-critical neighbor (NNN), each actor periodically determines its criticality (i.e., cut vertex or not). In addition, they both maintain two-hop neighborhood information. In order to minimize the message overhead, DCR [17] identifies the critical nodes with one-hop neighborhood information, and the restoration is similar to DARA's.

Since cut vertex identification incurs significant overhead in terms of messaging and state maintenance, RIM [35] does not distinguish the importance of nodes. All the one-hop neighbors move towards the position of the failed node till the distance is "$R_c/2$". Other nodes perform a cascade inward movement to connect to the connected network when they cannot communicate with the moved nodes. RIM is simple and efficient, especially for sparse networks. But the performance degrades for dense networks. RIM involves too many unnecessary motions, especially the first step. As all the failed node's neighbors do not know each other, they all move to the position which is "$R_c/2$" far from the failed node to maintain neighbor connectivity even when they are within half of the communication radius. These may incur outward motions. In addition, the message overhead is very high.

The above methods mainly focus on the moving distance and message overhead. In fact, network lifetime is the most important factor which depends on energy efficiency and load balancing. Abdelmalek [4] proposed a two-phase restoration algorithm. It searches the redundant nodes using the cluster heads, then restores connectivity, and energy consumption is taken into consideration. CoRF [3] is another connectivity restoration algorithm that strives to increase network lifetime. It selects a backup node according to the fuzzy logic rules. In addition, there are some realistic connectivity restoration methods [29] that take obstacles or terrain elevation into consideration. The direct path movement may be impossible, or is not optimally energy-efficient. In summary, energy efficiency and load balancing are the important evaluation metrics for the connectivity restoration algorithms.

4 Hybrid connectivity restoration algorithm

In this section, a hybrid connectivity restoration (HCR) algorithm is proposed to restore connectivity in WSANs. HCR is a distributed, localized, and efficient approach aiming at minimizing the cost of moving nodes. Instead of identifying a cut vertex, HCR just identifies the critical nodes. Each node maintains a one-hop neighborhood table including unique node ID (*ID*), local position (*POSITION*) and critical character (*CRITICAL*). HCR combines proactive backup node selection and reactive cascade node motion.

4.1 Proactive backup node selection

In order to minimize the number of messages and shorten restoration time, each node will select a backup node from its neighbors before a node failure occurs. During the initialization of a one-hop neighborhood table, each node sends a broadcast message containing its (*ID*) and (*POSITION*). All the nodes in its communication range will receive the message.

After a round of information exchanging, each node will determine whether it is critical through its one-hop neighborhood table NT. A node is an uncritical node if and only if all its one-hop neighbors form a connected network. In Fig. 2, node $A3$ has three one-hop neighbors $A1$, $A2$, and $A4$. They form two disjoint components $\{A1, A2\}$ and $\{A4\}$. So $A3$ is a critical node. Similarly, $A4$, $A5$, $A6$, and $A7$ are critical nodes. While node $A8$ is an uncritical node for all its one-hop neighbors $A5$, $A7$, $A9$ form a connected network $\{A5, A7, A9\}$, so as node $A9$. Node $A10$ is a leaf node and just has one one-hop neighbor $A7$, so it is also an uncritical node. It is worth mentioning that a critical node's failure will not necessarily divide a network, e.g., $\{A5, A6, A7\}$. However, an uncritical node's failure must not break network connectivity. Therefore, a backup node needs to be selected only for a critical node.

Different from DARA [1] and DCR [17] which select a backup node based on distance and node degree, HCR selects a backup node based on the minimum moving cost. The minimum moving cost of all backup nodes should also be the failure cost of the failure node. For a critical node i, choose any node j in its one-hop neighborhood table $NT(i)$, compute *BestPosition* j' it should move to so as to connect all the rest nodes in $NT(i)$. As shown in Fig. 3, node $A3$ has three one-hop neighbors $A1$, $A2$, and $A4$ which form two disjoint components $\{A1, A2\}$, and $\{A4\}$. The best positions for $A1$, $A2$, and $A4$ are $A1'$, $A2'$, and $A4'$. They are all in the communication range of the rest sibling nodes. $A1'$ and $A2'$ are on the boundary of the communication range of $A4$. $A4'$ is the intersection of the communication boundary of $A1$ and $A2$. The moving cost for node $A1$, $A2$, and $A4$ are $A1A1'$, $A2A2'$, and $A4A4'$. The node with the minimum moving cost will be selected as the backup node for node $A3$. Here, node $A2$ is selected as the backup node and *BestPosition* is $A2'$. The best position for relocating the backup node is shown in Fig. 4 and the pseudo-code for backup node selection is detailed in Algorithm 2.

In order to find a backup node and *BestPosition* for node i, compute the best position j' for each node $j \in NT(i)$ so that $d_{j,j'}$ is minimum and satisfies Eq. 1. $d_{j,j'}$ is the distance between nodes j and j', it is the moving cost of node j for failure node i. For all $j \in NT(i)$, the minimum $d_{j,j'}$ should be the failure cost of node i, and node j is elected as a backup node and j' is *BestPosition*.

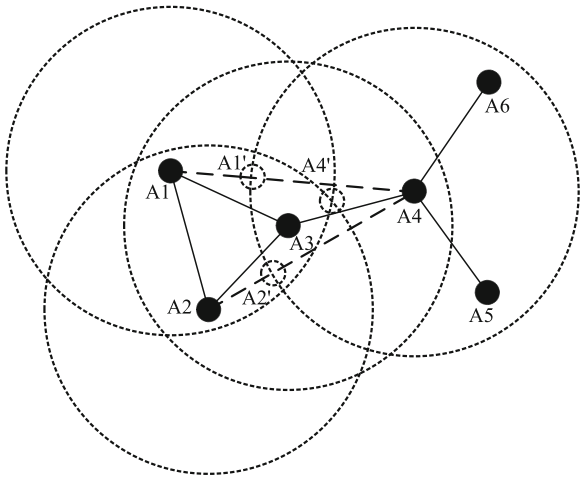

Fig. 3 An example of backup node selection

$$\forall k \in NT(i),\ k \neq j, \quad d_{k,j'} \leq R_c \tag{1}$$

Theorem 1 *BestPosition for node i must be the intersection of communication circle of any two nodes in $NT(i)$ or the intersection of the line ij and communication circle of node j.*

Proof First, the intersection of a pair of nodes' communication boundary in $NT(i)$ must connect these pair nodes. If this intersection is within the rest sibling nodes' communication range, this node can replace the failure node to connect all the failure node's neighbors. But the intersection may not be the optimal solution. Actually, the intersection is optimal only when the moved node is beyond the communication range of the pair of nodes. As shown in Fig. 4b, node C is out of the communication range of A and B. The intersection C_{new} should be the optimal position. But in Fig. 4a, node C is within the communication range of A. Node I is the intersection of A's and B's communication boundary. Node I can connect A and B, node C_{new} can also connect A and B, and $d_{C,C_{new}} < d_{C,I}$. $d_{C,C_{new}} + d_{C_{new},B} < d_{C,I} + d_{I,B}$ for Line Axiom, where $d_{C_{new},B} = d_{I,B} = R_c$. Thus I is not the optimal position, but node C_{new} is. □

Lemma 1 *The node failure cost is no more than the nearest-neighbor distance in HCR.*

Proof By Theorem 1, *BestPosition* must be the intersection of any pair of nodes' communication boundary in $NT(i)$ or the intersection of the two nodes connection line and one node's communication boundary. As shown in Fig. 4, node C_{new} is *BestPosition*. In Fig. 4a, $\angle CC_{new}F$ must be an obtuse angle; otherwise, FC_{new} will be the tangent line of circle B, and node F will be out of the circle. As $\angle CC_{new}F$ is an obtuse angle, $d_{C,C_{NEW}} \leq d_{C,F}$, so as Fig. 4b. Since each node's best position moving cost is less than its distance to the failure node, the node failure cost is no more than the nearest-neighbor distance. □

The previous approaches such as DARA and DCR all move the backup node to the position of the failure node, then the failure cost is equal to the nearest-neighbor distance in the best case. So HCR outperforms DARA and DCR in terms of travel distance in a motion. The shorter distance it moves, the smaller impacts on its neighbors' connectivity.

If there are two or more neighbor nodes with the same moving costs to restore the network connectivity, the one with the smallest failure cost will be selected as the backup node for the next restoration loop with the smallest moving cost. After selecting the backup node and obtaining *BestPosition*, it will send a broadcast message containing the information of the backup node and *BestPosition*. After a round of broadcasting, each node will update its backup node and *BestPosition* again and inform the changes to its new backup node only. By now, proactive backup node selection is finished. Then the node starts to

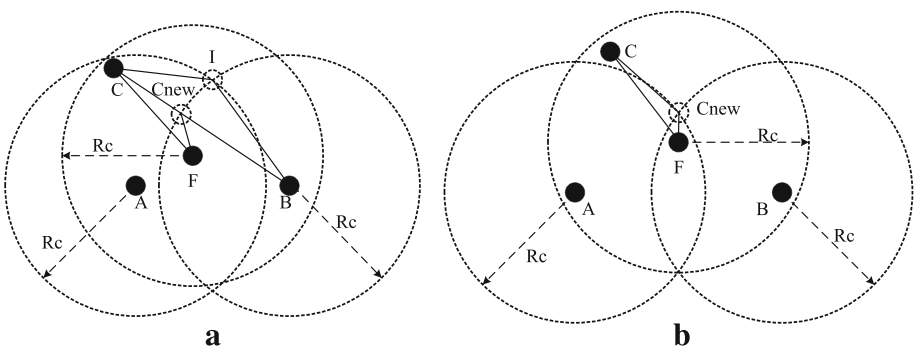

Fig. 4 Best position for relocating the backup node. **a** Node C is out of the communication range of *A* and *B*. **b** Node *C* is within the communication range of *A* and out of *B*

send heartbeat messages to the backup node periodically to declare that it is functional. Once the backup node does not receive the heartbeat message within a period, it will start reactive cascade node motion to restore connectivity.

4.2 Reactive cascade node motion

As mentioned before, only a critical node will break network connectivity, so when a critical node i fails, its backup node j will detect the failure the first time and trigger reactive cascading node motion. Before the backup node j moves to *BestPosition* j', it will update its one-hop neighborhood table $NT(j)$. It replaces the failure node's position by j', and checks whether its one-hop neighbors' connectivity is broken. If broken, node j will be the failure node and the restoration will be triggered. Node j will update its backup node k and *BestPosition* k'. Node k is selected from its rest neighbors except for the failed node i in order to avoid falling into an infinite loop. The backup node k will move to *BestPosition* k' to connect all its neighbors in $NT(j)$ and j'. After selecting the backup node k and *BestPosition* k', node j will send a message to backup node k about *BestPosition* k' it should move to and this will trigger a new round motion of node k. Node j will move to j' to replace the failure node i. This process will be repeated recursively until the failure node's neighbors are connected.

Figure 5 is an example for HCR cascading node motion. In Fig. 5a, node $A3$ fails. Its three one-hop neighbors $A1, A2$, and $A4$ become three isolated nodes and they are out of each other's communication range. For proactive backup node selection, node $A4$ is selected to move towards $A4'$ to connect $A1$ and $A2$. Due to $A4$'s movement, its neighbor $A6$ is out of the communication range, while $A5$ is still within the communication range in Fig. 5b. Before moving to $A4'$, node $A4$ selects a backup node to replace it and node $A6$ is selected. In Fig. 5c, node $A6$ moves to $A6'$ to connect node $A4'$ and node $A5$. The movement does not break its connection to node $A7$, and the whole network is connected. It is worth mentioning that $A4'$ is the intersection of $A1$'s and $A2$'s communication circles, while $A6'$ is the intersection of line $A4'A6$ and $A5$'s communication circle.

Algorithm 1 HCR

actor_coord: the coordinate of actors
R_c: the communication range for each actor
Initialize 1-hop neighborhood table *NEIGH_TABLE* for each actor;
for every actor i in a WSAN **do**
  if Is_Critical(i) **then**
    Algorithm Backup Node Selection
  else
    back_ID = 0; *BestPosition* =[*inf*, *inf*]; *failed_cost* = 0
  end if
end for
Each actor broadcasts a message about its failed cost *failed_cost*
for every actor i **do**
  if Is_Critical(i) and actor i has two neighbors **then**
    the actor with smaller *failed_cost* is selected as a backup node for i
  end if
end for
Every actor sends a message to notify its backup node to monitor its status and *BestPosition*
while an actor node B detects a failure of its neighbor F OR receives a movement message **do**
  node B adds *BestPosition* into its *NEIGH_TABLE*
  while Is_Critical(B) **do**
    Algorithm Backup Node Selection
    node B sends a movement message to its new backup node about *BestPosition*
    node B moves to *BestPosition* to replace the failed node F
  end while
end while

The pseudo-code for HCR is shown in Algorithm 1. HCR is a hybrid method that selects a backup node for each actor before it fails. At the beginning of network construction, each node will broadcast a message to notify its position within its communication range, and records

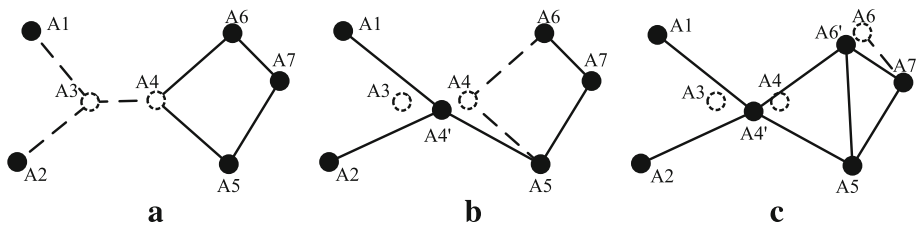

Fig. 5 An example HCR cascading node motion. **a** Node *A3* failed and node *A4* is selected to move. **b** Node *A4* moves to *A4'*. **c** Node *A6* moves to *A6'*

its neighbors' *ID* and *Position* to build one-hop neighborhood table *NEIGH_TABLE* (line 3). Then each node will identify whether it is critical through its *NEIGH_TABLE*. For an uncritical node, its failure will not break connectivity, so *failed_cost* $= 0$ (line 8). While for a critical node, it will select a backup node and *BestPosition* (line 6). The algorithm for backup selection is detailed in Algorithm 2. When a node just has two neighbors, move any one towards another has the same travel distance to restore connectivity. For this case, the one with smaller *failed_cost* is chosen (line 11-16), this is to insure the next restoration has a smaller moving cost. By now, the initialization and proactive backup selection are done. Every actor sends a message to notify its backup node to monitor its status and *BestPosition*. When a node detects a failure of its neighbors, it will trigger the cascading node motion (line 18-25). The backup node will add *BestPosition* into its *NEIGH_TABLE* to tell whether it is critical. If so, it will select its new backup node before moving to replace the failed node (line 21). Otherwise, network connectivity is restored.

Algorithm 2 Backup Node Selection

1: R_c: the communication range for each actor
2: *NEIGH_TABLE*: the failed node's one-hop neighborhood table
3: F: the failed node
4: F': *BestPosition* the failed node moved to
5: **for** every actor i except F' in *NEIGH_TABLE* **do**
6: **for** every actor j in *NEIGH_TABLE* **do**
7: Compute the intersection of circle j and line L_{ij}
8: Add the intersection into *Candidate_Set*
9: **for** every actor k in *NEIGH_TABLE* and $k \neq j$ **do**
10: Compute the intersection of circle j and circle k, where the radius is R_c
11: Add the intersection into *Candidate_Set*
12: **end for**
13: **end for**
14: **for** every node m in *Candidate_Set* **do**
15: **for** every actor n in *NEIGH_TABLE* **do**
16: **if** $dm, n > R_c$ **then**
17: remove the node m from *Candidate_Set*
18: **break;**
19: **end if**
20: **end for**
21: **end for**
22: Choose the closest point in *Candidate_Set* from node i as its *BestPosition*
23: **end for**
24: Return the node's *ID* and *BestPosition* which has the minimum *travel_dist* in *NEIGH_TABLE*

The backup node selection is the key part of HCR and the pseudo-code is presented in Algorithm 2. It is worth mentioning that all the selection of backup node is done on the failed node or moved node and it just explores *NEIGH_TABLE*. During this process, it does not need to send message to other nodes. This reduces the message overhead. According to Theorem 1, *BestPosition* must be the intersection of any pair of nodes' communication boundary or the intersection of the two nodes connection line and one node's communication boundary. Firstly, for each node i in *NEIGH_TABLE*, compute its *Candidate_Set* (line 6-13). Then remove the position that is out of the rest nodes' communication range (line 14-21). Afterwards, choose the closest point in *Candidate_Set* as *BestPosition* for node i. Finally, return the node's *ID* and *BestPosition* which has the minimum *travel_dist* in *NEIGH_TABLE*.

5 Algorithm analysis

HCR combines proactive and reactive methods to handle network connectivity restoration from a single-node failure in WSANs. The selection of a backup node is proactive, while the restoration is reactive. This scheme shortens the restoration process and reduces the overhead including distance cost and message cost. Next, the performance of HCR is analyzed.

First and foremost, network connectivity after a single-node failure is restored. It is assumed that no other node fails during the restoration process and any two nodes can communicate with each other directly if they are R_c apart or closer. Then network connectivity is not weakened and no new critical node is introduced during the restoration. In addition, load balancing is taken into consideration and no node travels too far while others too close. The overhead of communication and the complexity of computing are also analyzed. We introduce the following theorems.

Theorem 2 *HCR restores network connectivity after a single-node failure.*

Proof Uncritical node failure will not break network connectivity since all its neighbors are connected when it is removed from the network. HCR identifies a critical node at the initialization time and selects a backup node and decides *BestPosition*. When a critical node fails, it will trigger the restoration, and its backup node moves to *BestPosition* to reconnect all its one-hop neighbors. In the following cascading node motion, the moved node will reconnect its sibling nodes until all its siblings are connected. In order to avoid an endless loop, each node can only move once during the restoration, and a moved node will not be selected as a backup node in the future. This can guarantee that HCR terminates in limited number of steps. □

Theorem 3 *The total number of critical nodes does not increase.*

Proof In HCR, the selection of a backup node and *BestPosition* guarantees that the total number of critical nodes does not increase. The critical of backup node will change into that of the failed node. And the degree of each node will not decrease. The backup node will move to *BestPosition* to connect all its siblings. In Fig. 6, node $A3$ fails, and it selects a neighbor to reconnect all its neighbors. In Fig. 6a, node $A1$ is selected and moves to node $A1'$ where it is on the line $A1A4$, and $d_{A1',A4} = R_c, d_{A1',A2} < R_c$. Since $A1$ and $A2$ can connect before failure, we just need to guarantee that node $A1'$ is in the communication range of $A2$. In Fig. 6b, node $A4$ is selected and moves to node $A4'$ where it is the intersection of circle $A1$ and $A2$, $d_{A1,A4'} = R_c$, and $d_{A2,A4'} = R_c$. Node $A4$ is out of reach by $A1$ and $A2$ before the failure. After restoration, the unmoved critical nodes are not changed, e.g., $A2$ and $A4$ in Fig. 6a, and the moved uncritical node becomes a critical node or not. Node $A1'$ is critical, where node $A4'$ is uncritical. Before the restoration, the failed node is critical, so the total number of critical nodes does not increase. □

Theorem 4 *The maximum distance a node travels in HCR is the communication range R_c.*

Proof In HCR, a backup node is one of the failed node's neighbors, and it moves to *BestPosition* to reconnect all its siblings. In the worst case, a backup node moves to the position of the failed node, and it must connect all its siblings. That is the relocated scheme in DCR [17]. It has been proven that the maximum distance a node travels in DCR is the communication range R_c. Lemma 1 proves that HCR outperforms DCR in terms of travel distance in one motion.

Take Fig. 7 as an example. Node $A3$ is the failed node, and node $A4$ moves to $A4'$ to connect $A1$ and $A2$, and $d_{A4,A4'} < d_{A4,A3} < R_c$. Due to the motion of node $A4$, node $A5$, and node $A6$ will be out of reach by $A4'$, so before moving to $A4'$, $A4$ will select a node from its one-hop neighbors $\{A5, A6\}$. Suppose $A6$ is selected and *BestPosition* is $A6'$, then $d_{A6,A6'} < d_{A6,A4} < R_c$. □

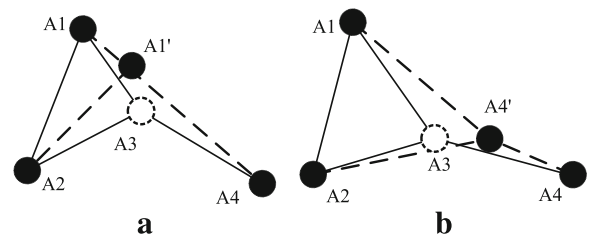

Fig. 6 No new critical node is introduced during the restoration. **a** Node *A1* is selected to move. **b** Node *A4* is selected to move

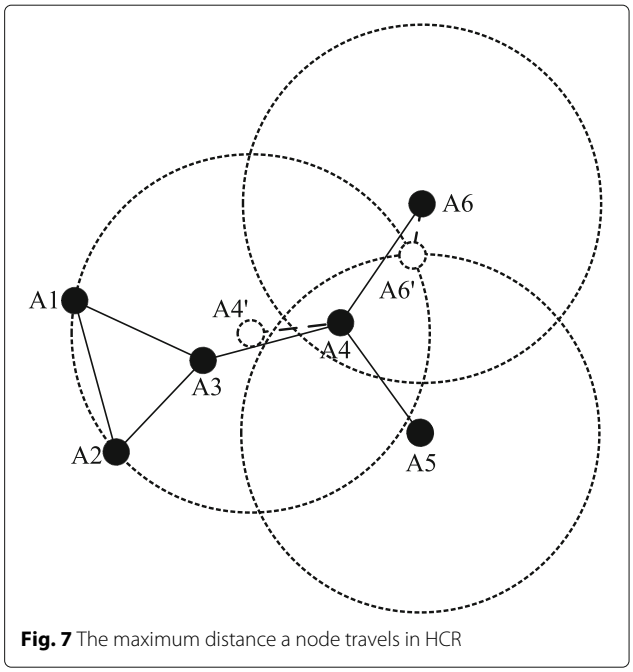

Fig. 7 The maximum distance a node travels in HCR

Theorem 5 *The shorter distance a node travels, the fewer nodes are affected. The probability of a node affected by its moving neighbor is $(1 - \frac{2\theta - \sin 2\theta}{\pi})$, where $\theta = \arccos d/2R_c$, d is the travel distance, and R_c is the communication range. It approximately equals 0.62 times of d/R_c.*

Proof Take Fig. 8a as an example. Node A moves to A'. The nodes in the shaded area like node B are within the communication range of node A, while out of reach by A'. The travel distance equals d. It is assumed that the nodes are deployed randomly, so a node has an equal chance to reside at any place. The probability of a node affected by its moving neighbor is the probability that it is located in the shaded area. Therefore, the probability is the shaded part area divided by the whole communication area. According to symmetry and area formula of a sector, the shaded part area equals $\pi {R_c}^2 - (2\theta \cdot {R_c}^2 - \sin 2\theta \cdot {R_c}^2)$, where $\theta = \arccos d/2R_c$. So the probability of a node affected by its moving neighbor is $(1 - \frac{2\theta - \sin 2\theta}{\pi})$. In Fig. 8b, the solid line reflects the probability against d/R_c, where the dashed line reflects the growth rate of probability. It shows that the growth rate is approximately 0.62. □

Theorem 6 *The time complexity of the backup node and BestPosition selection is $O(n^3)$, where n is the number of failed node's one-hop neighbors.*

Proof It has been proven that *BestPosition* must be the intersection of any pair of nodes' communication boundary in $NT(i)$ or the intersection of two nodes connection line and one node's communication boundary in

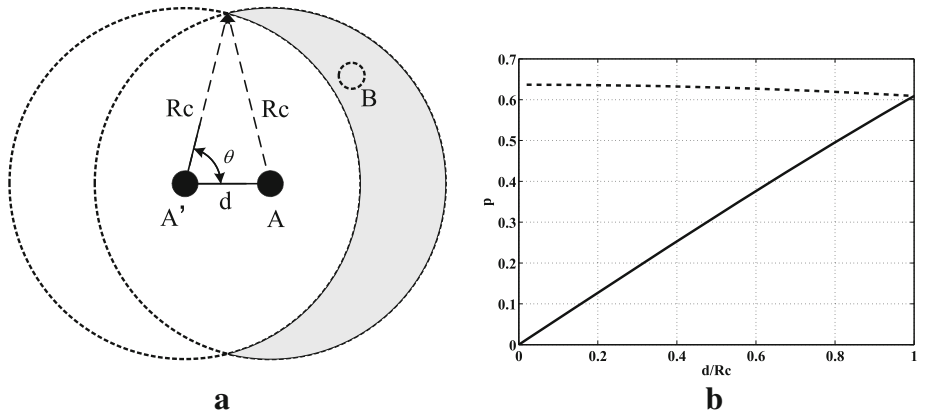

Fig. 8 The influence of node moving distance. **a** An example of the influence of node moving. **b** The influenced probability of a node against *d/Rc*

Theorem 1. For each node, the intersection of any pair of its sibling nodes' circle is computed firstly, and the time complexity is $O(n^2)$. Then the intersection of a line that connects this node to its sibling node and the sibling node circle is obtained, and the time complexity is $O(n)$. Finally, check whether the intersection is within the communication range of all the sibling nodes and find the minimum travel distance. The time complexity is also $O\left(n^2\right)$. Therefore, the time complexity of the best position selection for each node is $O\left(n^2+n+n^2\right)$, that is $O\left(n^2\right)$. Then the node with the minimum traveling distance will be selected as the backup node, and its best position will be *BestPosition*. Hence, the time complexity of the backup node and *BestPosition* selection is $O\left(n^3\right)$. □

Theorem 7 *The total message complexity of HCR is O(N), where N is the number of actors.*

Proof The selection of a backup node and *BestPosition* is done at the failed or moved nodes in HCR. It just maintains one-hop neighborhood table for each node. In addition, the failed node only sends a message to its backup node about its movement. In the worst case, there are $N-2$ nodes moving. Each moved node sends a movement message to its backup node. Therefore, the total message complexity of HCR is $O(N)$, where N is the number of actors. It is worth noting that the exchange with neighbors at a new position does not count in HCR, and it is considered as a part of status update for maintaining one-hop neighborhood table. □

Theorem 8 *The time it takes HCR to restore network connectivity is proportional to N and R_c, where N is the number of actors and R_c is the communication range.*

Proof Firstly, HCR proactively selects a backup node and *BestPosition* before a node failure, so when a node fails, the backup node will move to *BestPosition*. This will trigger cascading motion. For each moved node, it will select its new backup node and *BestPosition* before its moving and send a message to notify its backup node to move to *BestPosition*. According to Theorem 7, the time complexity is $O\left(n^3\right)$ where n is the number of its neighbors. Usually, n is very small compared with the total number of nodes N, so the computing time can be ignored. In the worst case, there are $N-2$ nodes moving, and each node moves at most R_c, so the total time it takes HCR to restore network connectivity is $(N-2)\times R_c$, which does not exceed $(N\times R_c)$. □

6 Simulation results

Extensive simulations have been conducted to evaluate the performance of the proposed HCR compared with the previous algorithms RIM and DCR. The simulation settings and performance metrics are introduced in Section 6.1, and the detailed results and analysis are presented in Section 6.2.

6.1 Simulation settings and performance metrics

In order to evaluate the performance of HCR and compare it with the previous algorithms RIM [35] and DCR [17] fairly, we carry out all the simulations on Matlab R2012a with an Intel Core i3-3220 CPU and 8 G RAM computer. In the simulations, numerous mobile actors are deployed randomly in an area of 1000 m×1000 m. All the actors are homogeneous with the same communication range. The energy cost and lifespan are pursued during the connectivity restoration in varied applications, but the actual energy cost is difficult to model and capture during simulations, so the following four metrics are employed to evaluate the performance of HCR:

- *Number of relocated nodes*: It reports the average number of relocated nodes during a single-node

failure restoration. This metric assesses the scope of connectivity restoration within a network.

- *Total distance moved*: It reports the total distance that the involved nodes move during the restoration. This metric assesses the efficiency of the restoration methods from the standpoint of a network.
- *Average travel distance*: It depicts the average distance that the involved nodes travel during the restoration. This metric assesses the efficiency standing in the perspective of a node.
- *Number of messages*: It captures the total number of messages sent among the nodes during the restoration. This metric assesses the communication overhead of the restoration methods.

In addition, to study the impact of network topology on the performance of HCR, the network topology is varied in different simulations with the following parameters:

- *Number of deployed nodes* (*DN*): It reports the number of the deployed nodes in an area. Since the area is 1000 m × 1000 m, this parameter actually represents the density of a WSAN. The larger the *DN*, the bigger the node density, indicating a stronger network connectivity. In a rather highly connected WSAN, a node has many one-hop neighbors, and it is easier to choose a backup node and *BestPosition*. It will increase the time of selecting at the same time.
- *Communication range* (R_c): It is assumed that all the actors are homogeneous with the same communication range and a pair of nodes can communicate with each other when they are within each other's communication range. R_c also affects network density *DN*. Small R_c will generate a rather sparse network, while large R_c will increase network connectivity. This will also increase the travel distance of the involved nodes during the restoration under HCR.

6.2 The overall results

Both sparse and dense networks with different values of *DN* and R_c are simulated to evaluate the performance of HCR compared with RIM [35] and DCR [17]. The value of *DN* is chosen from set {10, 20, 40, 60, 80, 100} and the value of R_c is chosen from set {20, 40, 60, 80, 100, 120}. For every topology, 10 WSANs are generated, and for every WSAN, 30% of the deployed nodes are randomly selected forming the failed node set. For every WSAN, RIM, DCR, and HCR are run after a node fails and the average values are computed. The overall results are shown in Table 1. We can see that HCR outperforms RIM and DCR in terms of the number of relocated nodes, total travel distance, average travel distance, and number of messages for both high-density and low-density networks. The results and analysis are detailed in the following.

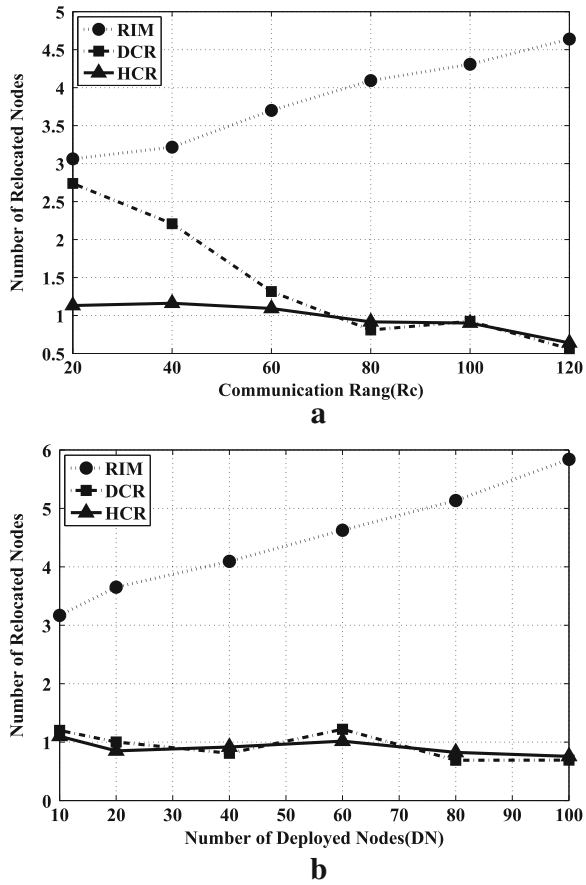

Fig. 9 The number of relocated nodes during the restoration while varying **a** the communication range (with $DN = 40$) and **b** the network size (with $R_c = 80$)

6.2.1 Number of relocated nodes

Figure 9a, b reports the number of the relocated nodes during the restoration under various communication range R_c and network size *DN*, respectively. The plotted results are the average over multiple independent simulations. For any topology, 30% of *DN* fail randomly. The two figures indicate that DCR and HCR relocate fewer nodes than RIM. This is because RIM requires all the neighbors of the failed node to move inward. In fact, sometimes they may move outward to the position that is $R_c/2$ away from the failed node. In addition, the number of the relocated nodes grows with the increasing of R_c and *DN* since the larger R_c and *DN*, the more neighbors a failed node has. DCR and HCR have the same results when R_c and *DN* increase, because they both identify critical nodes, and only move one of the neighbors to replace the failed node in a round of restoration. It is worth noting that the candidate selection is processed at the failed node before it fails or moves, so they just maintain one-hop neighborhood table.

Table 1 Overall simulation results

DN	RC	Number of relocated nodes			Total moved distance			Number of messages			Average travel distance		
		RIM	DCR	HCR	RIM	DCR	HCR	RIM	DCR	HCR	RIM	DCR	HCR
10	20	2.5000	0.8667	0.7667	9.4481	11.7579	4.0453	11.2667	0.8000	0.6000	3.7173	6.4224	2.3910
	40	2.5333	1.1667	1.2000	22.3878	34.1477	14.3315	10.3333	1.1333	1.2000	8.5230	16.8824	6.6039
	60	2.7000	1.2667	1.2000	34.3570	54.9098	20.9049	11.3333	1.2667	1.1333	11.8734	27.7736	11.4697
	80	3.1667	1.2000	1.1000	54.5616	68.3552	25.6540	15.0000	1.2667	1.0667	16.9988	30.8971	12.3777
	100	2.8667	0.5667	0.8667	66.9697	41.0099	26.1766	13.0000	0.2000	0.8000	21.3045	33.2997	13.1408
	120	2.7333	0.9333	1.0333	71.1571	80.9414	35.6322	12.6000	0.8000	1.0000	27.5052	44.9467	17.3013
20	20	2.6667	1.1500	1.0167	10.1203	15.8547	5.7045	12.9334	1.1333	0.8667	3.6459	8.0768	3.2332
	40	2.9500	2.2167	1.2000	24.3734	64.1113	14.8084	13.3000	3.0000	0.9667	8.0881	20.7230	8.5739
	60	3.0333	1.8667	1.2000	38.7232	78.6362	22.8659	14.4000	2.4000	1.0667	11.9809	27.1874	12.7959
	80	3.6500	1.0000	0.8500	60.9370	57.9764	19.7107	20.6000	0.9000	0.6000	15.7861	31.1569	13.0692
	100	3.7833	0.6833	0.8000	80.4082	47.1660	22.1815	21.8000	0.4000	0.6333	21.5298	33.2313	13.5711
	120	3.7667	0.8833	0.8000	101.2945	72.8401	21.3106	22.7333	0.7333	0.5667	25.8660	42.7639	13.5360
40	20	3.0615	2.7385	1.1308	11.9039	37.9415	6.3112	15.3077	4.0923	0.8769	3.7377	9.4159	3.6941
	40	3.2154	2.2077	1.1615	25.6699	61.7150	11.2814	16.4154	3.0308	0.9385	7.7238	19.2907	6.6423
	60	3.7000	1.3154	1.0923	49.2846	58.5885	20.8707	20.4000	1.4154	0.9692	13.1892	26.9827	10.9955
	80	4.0923	0.8077	0.9154	69.0521	42.6712	18.4933	25.7077	0.5077	0.7231	16.7857	28.9037	10.8412
	100	4.3077	0.9231	0.9000	91.6229	61.6710	19.9848	27.9231	0.7846	0.7385	20.0957	33.9756	11.3504
	120	4.6385	0.5615	0.6385	113.5523	45.2215	20.8300	35.2000	0.3385	0.4923	24.4107	30.6194	12.0742
60	20	3.4300	3.4300	1.4600	13.7143	47.4912	8.3110	17.1500	5.1700	1.2300	3.9234	11.2197	4.6324
	40	3.7250	2.2900	1.2100	30.3220	66.0921	12.9398	20.1000	3.1700	1.0100	7.9336	20.0784	7.1632
	60	4.5600	1.3700	1.1050	59.5532	59.2711	17.5455	28.9300	1.5200	0.9900	12.7306	25.7413	9.1642
	80	4.6250	1.2200	1.0150	76.5166	67.7799	21.1872	30.6400	1.2200	0.8100	16.7286	32.0698	12.1174
	100	5.7300	0.5450	0.6150	117.2685	34.8617	14.5538	44.9600	0.2800	0.4200	20.5859	24.7201	9.4990
	120	6.8150	0.4950	0.5450	167.6632	39.0700	16.8272	67.3100	0.2500	0.3500	24.6528	28.8812	11.4855
80	20	3.0808	4.1846	1.3654	11.8660	59.3107	7.6236	15.0539	6.8077	1.1692	3.8286	10.6448	4.0656
	40	3.7692	1.3269	1.1769	31.2300	37.8081	13.7661	20.8461	1.3538	1.0538	8.0497	18.1350	7.2869
	60	4.3308	0.9231	0.8731	53.2154	36.9789	13.5749	29.1923	0.8000	0.7000	12.1273	20.0345	7.9371
	80	5.1346	0.6885	0.8231	86.9678	36.2184	15.8676	38.2769	0.3308	0.6000	17.1423	26.8037	9.7161
	100	5.7808	0.6192	0.7385	118.1022	41.5090	15.8088	46.8615	0.3385	0.5769	20.1898	29.5200	9.1799
	120	6.3769	0.5077	0.5846	155.5264	38.9712	14.9357	58.1000	0.2923	0.4462	23.9728	27.1466	9.5740
100	20	3.3333	2.4485	1.2576	13.0345	33.3660	6.4384	17.0909	3.3879	1.0061	3.7086	10.0852	3.6431
	40	3.9000	1.4879	1.0394	32.7427	41.4879	11.4844	22.4606	1.7030	0.8061	8.2931	17.3802	6.9119
	60	5.0667	0.8970	0.8152	63.4889	37.4344	13.0444	35.6909	0.8000	0.6364	12.2978	20.0826	7.8285
	80	5.8394	0.6909	0.7545	98.9894	36.4659	14.7784	48.1454	0.4121	0.5394	16.8274	24.5749	9.2227
	100	6.8515	0.6061	0.6970	140.9332	39.3920	15.3282	66.2667	0.3697	0.5515	20.7653	26.3370	8.7655
	120	8.4636	0.4000	0.5182	201.1593	29.5398	13.5253	92.2545	0.1333	0.3697	23.6561	23.8265	8.8305

Moreover, Fig. 9a shows that HCR outperforms DCR when the communication range is less than 72, and has the similar results when the communication range grows. This is because of the candidate selection. HCR will choose a node to move the least distance to *BestPosition* and to reconnect the failed node's neighbors, while DCR chooses the backup node by critical, degree, and distance. Furthermore, DCR moves the backup node to the failed node, and the failed node is not at *BestPosition* most of the time. When R_c is small, the network is sparse, and the choice of a backup node is too limited, so HCD outperforms DCR. When R_c is large, the network is dense, and it is likely to choose an uncritical node to restore network connectivity, so DCR can get the same result with HCR. Figure 9b shows that *DN* has little impact on HCR and DCR in terms of relocated nodes when $R_c = 80$. At the moment, the WSAN is dense for the communication range. Considering Fig. 9a, b, we can see that R_c is more influential than *DN* on the number of relocated nodes.

6.2.2 Total moved distance

The total moved distance is an important metric for network connectivity restoration, since the actors are energy-limited while moving is an energy-consuming operation. As shown in Fig. 10a, b, HCR outperforms RIM and HCR for any R_c and *DN*. As can be seen in Fig. 10a, b, RIM grows when R_c or *DN* increases, and it increases linearly. This is terrible and unacceptable when the network scale is large. While the curve of DCR is unstable, it outperforms RIM when the network is dense. In the simulations, when $DN = 40$ and $R_c > 62$, DCR outperforms RIM. Similarly, when $R_c = 80$ and $DN > 20$, DCR outperforms RIM. At the same time, HCR remains stable when varying R_c or *DN*.

In Fig. 10a, the curve of HCR rises slowly when $R_c < 60$, then it remains stable. It decreases with the increase of *DN* in Fig. 10b. When the communication range is small, the network is sparse and the failed node has limited neighbors to select, so HCR needs many nodes to move to restore network connectivity. Since each node just needs to move a little, the growth is slow and small. While when the communication range increases, many choices make a rapid convergence. It is the same for DCR. However, DCR moves the backup node to the failed node, while HCR moves the backup node to *BestPosition*. It has been proven in Lemma 1 and the simulation results also verify that HCR outperforms DCR in terms of total moved distance. Figure 10b shows a decreasing of the total moved distance for HCR when increasing *DN*. This is because when R_c is determined, the growing of *DN* will increase the choices, and moving a shorter distance is enough to restore network connectivity.

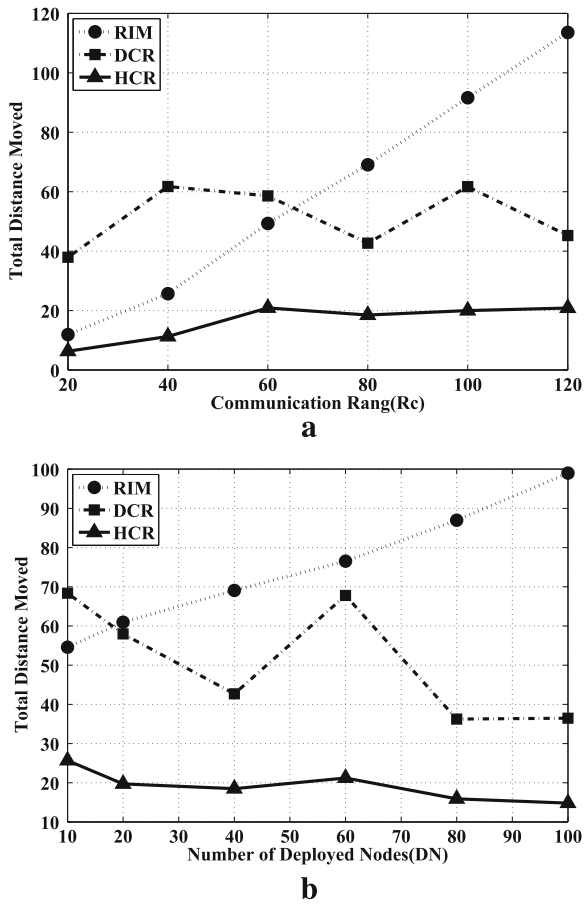

Fig. 10 The total moved distance during the restoration while varying **a** the communication range (with $DN = 40$), and **b** the network size (with $R_c = 80$)

6.2.3 Average travel distance

As aforementioned, node motion is a high energy-consuming operation, and nodes are prone to be out of work due to energy depletion. A node failure will incur cascading motion and more energy consumptions creating a vicious spiral. So the average travel distance of the involved nodes is of great importance in assessing the connectivity restoration algorithms.

Figure 11a shows that the curves of DCR and RIM grow with the increase of R_c while $DN = 40$. HCR also grows until $R_c = 60$, then it maintains stable. RIM outperforms DCR since it moves more nodes, and the biggest distance of RIM is limited to $R_c/2$, while DCR moves the backup node to the position of the failed node, and each involved node moves more than that in RIM. But they both increase linearly, which is unbearable in large-scale WSANs. Since the backup node moves to *BestPosition* in HCR, the involved nodes will move less than DCR, and when the network is dense, it is stable to move some distance to restore connectivity. As shown in Fig. 11a, when

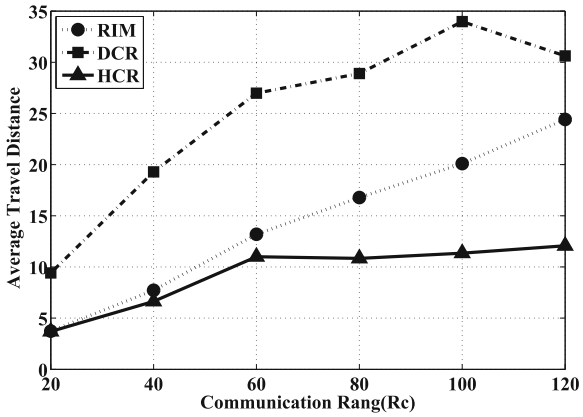

Fig. 11 The average distance a node travels during the restoration while varying **a** the communication range (with $DN = 40$), and **b** the network size (with $R_c = 80$)

$R_c > 60$, the average distance is around 11. RIM obtains the same results as HCR when the communication range is small.

Figure 11b shows very different results from Fig. 11a. The average travel distance for each involved node almost remains unchanged with varied network size when R_c is 80. This indicates that network size does not influence the average travel distance because each node moves at most $R_c/2$ in RIM, while R_c is fixed to be 80. Both the curves of HCR and DCR decrease when increasing the deployed nodes with $R_c = 80$. The WSAN is dense, thus the failed node has more neighbors which increases the probability of choosing the best replacement position of nodes. Due to moving the backup node to the position of the failed node, HCR moves less than DCR.

6.2.4 Number of messages

Communication cost is another important metric to assess the connectivity restoration methods because communication consumes large bandwidth resource which is very limited in WSANs. Figure 12a, b reports the total number of messages that need to be sent in restoring connectivity corporately. RIM incurs the highest message overhead since the decision of restoration is made by the failed node's neighbors in RIM. It needs to send the new position to its sibling nodes, so the communication cost is very lager. It becomes much worse when the density of a network increases because the number of neighbors grows.

Figure 12a, b indicates that HCR and DCR achieve similar message overhead when the network is dense. This is because the decision of restoration is made by the failed node. The communication is maintained just between the failed node and its backup node. Figure 12b also shows that the number of deployed nodes will influence the messages when the communication range equals 80. It is also true for other communication range, and the detailed results are shown in Table 1. It should be noted that HCR outperforms DCR when the communication range is small. In Fig. 12a, the curve of HCR is below that

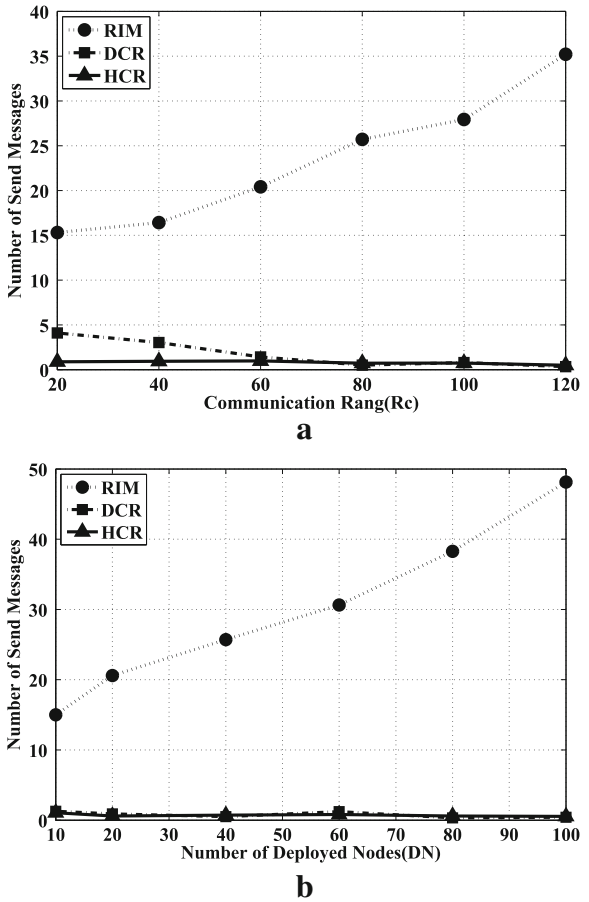

Fig. 12 The total number of messages during the restoration while varying **a** the communication range (with $DN = 40$) and **b** the network size (with $R_c = 80$)

of DCR when the communication range is less than 60. That means HCR outperforms DCR when the network is sparse. It contributes to the motion strategy. HCR moves less distance than DCR so that it incurs less motion in cascading moving. In sparse networks, the backup node's traveling distance has a great impact on the cascading moving. Therefore, HCR outperforms DCR since it moves less during a node's motion.

In conclusion, HCR outperforms RIM and HCR in terms of the four evaluation metrics on all the aspects whether the network is sparse or dense because of its proactive backup node selection and reactive cascading node motion. Though the proactive backup node selection is time consuming, it is carried out before the node failure. This will improve the response time. During the reactive cascading node motion, the computing of *BestPosition* is complex and time-consuming. While HCR is a distributed and localized method, the number of each node's neighbors is very small in all kinds of applications, so the selection of *BestPosition* will not spend so much time, and the connectivity restoration process will be fast.

7 Conclusions

There is a growing interest in the applications of WSANs in the recent years. Due to the harsh employed environment and limited energy supply, WSAN is prone to be out of work, which may break network connectivity. In this paper, we investigate the problem of restoring network connectivity when a single node fails. A hybrid distributed, localized, and efficient connectivity restoration algorithm HCR is proposed to solve this problem through moving the backup node to *BestPosition*. Compared with the previous schemes, HCR performs a localized network analysis to identify critical nodes, and only a critical node's failure triggers the restoration process. It is a compromised proposal between the cut vertex identification and non-identification. It is effective and has low complexity.

The performance of HCR is analyzed mathematically and validated through simulations. The simulation results have confirmed the effectiveness of HCR in terms of all the evaluation metrics. More importantly, HCR is applicable to various network topologies, sparse or dense. The performance of HCR remains stable when varying network topology. Though a comprehensive network will increase the complexity of the selection of *BestPosition*, it is acceptable.

Though HCR is designed for restoring network connectivity after a single-node failure, that means it can only deal with a single-node failure at a time and handle the sequential node failures. It can be extended to hand multi-node failures at a time by adding one more constraint that no two nodes share the same backup node. In addition, the investigated WSANs are two-dimensional, and we plan to study three-dimensional WSANs in the future. At the same time, coverage is another factor that can be taken into consideration in connectivity restoration, which is also our future research interest.

Acknowledgements
This research work was partly supported by the Special Project on Youth Science and Technology Innovation Research Team of Sichuan Province, under grant No.(2015TD0002) and Science and Technology Innovation Seed Project of Sichuan Province, under grant No.(2017RZ0008). We give thanks for the insightful discussion and help of Aiguo Chen. Finally, we give great thanks to the anonymous reviewers for the their suggestions to improve the quality of this paper.

Funding
This research work was partly supported by the Special Project on Youth Science and Technology Innovation Research Team of Sichuan Province, under grant No. (2015TD0002) and Science and Technology Innovation Seed Project of Sichuan

Authors' contributions
All authors contributed equally to this work. KY and GL contributed to the conception and design of the study. KY and LT contributed to the algorithms and simulation. KY, QJ and CP contributed to the analysis and interpretation of simulation data. All authors read and approved the final manuscript.

Authors' information
Ke Yan received the B.S. degree and M.S. degrees from the University of Electronic Science and Technology of China, in 2012 and 2015, respectively. He is currently pursuing the Ph.D. degree in the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China. His interests include wireless sensor network, network survivability, social network and privacy preserving.
Guangchun Luo received the B.S. degree, M.S. degrees and Ph.D. degrees in computer science from University of Electronic Science and Technology of China, Chengdu, China, in 1995, 1999 and 2004, respectively. He is currently a Professor of computer science at the University of Electronic Science and Technology of China, Chengdu, China. His research interests include computer networking, cloud computing, big data.
Ling Tian received the B.S. degree, M.S. degrees and Ph.D. degrees in computer science from University of Electronic Science and Technology of China, Chengdu, China, in 2003, 2006 and 2010, respectively. She is currently an Associate Professor with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu. She was a visiting scholar in Georgia State University in 2013. Her research interests include digital multimedia, cloud computing, big data.
Qi Jia received the B.S. degree and M.S. degrees from the University of Electronic Science and Technology of China, in 2013 and 2016, respectively. He is currently pursuing the Ph.D. degree in the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China. His interests include wireless sensor network, big data and privacy preserving.
Chengzong Peng received the B.S. degree from University of California, Irvine in 2017. He is currently pursuing the Master degree in the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China. His interests include mathematics, social network and privacy preserving.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Received: 12 May 2017 Accepted: 25 July 2017
Published online: 08 August 2017

References

1. AA Abbasi, M Younis, K Akkaya, Movement-assisted connectivity restoration in wireless sensor and actor networks. IEEE Trans. Parallel Distributed Syst. **20**(9), 1366–1379 (2009)
2. K Akkaya, F Senel, A Thimmapuram, S Uludag, Distributed recovery from network partitioning in movable sensor/actor networks via controlled mobility. IEEE Trans. Comput. **59**(2), 258–271 (2010)
3. U Baroudi, M Aldarwbi, in *Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on*. Corfl: A connectivity restoration mechanism using fuzzy logic in wireless actor and sensor networks (IEEE, Liverpool, 2015), pp. 1449–1454
4. A Boudries, M Amad, P Siarry, Novel approach for replacement of a failure node in wireless sensor network. Telecommun Syst. **65**, 341–350 (2017)
5. Z Cai, ZZ Chen, G Lin, A 3.4713-approximation algorithm for the capacitated multicast tree routing problem. Theor. Comput. Sci. **410**(52), 5415–5424 (2008)
6. Z Cai, R Goebel, G Lin, Size-constrained tree partitioning: Approximating the multicast k-tree routing problem. Theor. Comput. Sci. **412**(3), 240–245 (2011)
7. Z Cai, G Lin, G Xue, in *Proceedings of 11th Annual International Conference on Computing and Combinatorics 2005(COCOON)*. Improved approximation algorithms for the capacitated multicast routing problem (Springer, Kunming, 2005), pp. 136–145
8. M Cardei, J Wu, S Yang, Topology control in ad hoc wireless networks using cooperative communication. IEEE Trans. Mobile Comput. **5**(6), 711–724 (2006)
9. S Cheng, Z Cai, H Gao, J Li, Digging kernel component from big sensory data in wireless sensor networks. IEEE Trans. Knowl. Data Eng. **29**(4), 813–827 (2017)
10. S Cheng, Z Cai, J Li, Curve query processing in wireless sensor networks. IEEE Trans. Veh. Technol. **64**(11), 5198–5209 (2015)
11. S Cheng, Z Cai, J Li, X Fang, in *Computer Communications (INFOCOM), 2015 IEEE Conference on*. Drawing dominant dataset from big sensory data in wireless sensor networks, (Kowloon, 2015), pp. 531–539
12. SK Gupta, P Kuila, PK Jana, in *Proceedings of the Second International Conference on Computer and Communication Technologies*. Genetic algorithm for k-connected relay node placement in wireless sensor networks (Springer, New Delhi, 2016), pp. 721–729
13. N Haider, M Imran, NM Saad, MA Zakariya, in *Communications (MICC), 2013 IEEE Malaysia International Conference on*. Performance analysis of reactive connectivity restoration algorithms for wireless sensor and actor networks (IEEE, Kuala Lumpur, 2013), pp. 490–495
14. HA Hashim, B Ayinde, M Abido, Optimal placement of relay nodes in wireless sensor network using artificial bee colony algorithm. J. Netw. Comput. Appl. **64**, 239–248 (2016)
15. Z He, Z Cai, S Cheng, X Wang, Approximate aggregation for tracking quantiles and range countings in wireless sensor networks. Theor. Comput. Sci. **607**(4), 381–390 (2015)
16. M Imran, M Younis, AM Said, H Hasbullah, in *Embedded and ubiquitous computing (EUC), 2010 IEEE/IFIP 8th international conference on*. Partitioning detection and connectivity restoration algorithm for wireless sensor and actor networks (IEEE, Hong Kong, 2010), pp. 200–207
17. M Imran, M Younis, AM Said, H Hasbullah, Localized motion-based connectivity restoration algorithms for wireless sensor and actor networks. J. Netw. Comput. Appl. **35**(2), 844–856 (2012)
18. S Lee, M Younis, M Lee, Connectivity restoration in a partitioned wireless sensor network with assured fault tolerance. Ad Hoc Netw. **24**, 1–19 (2015)
19. J Li, S Cheng, Z Cai, J Yu, C Wang, Y Li, Approximate holistic aggregation in wireless sensor networks. ACM Trans. Sensor Netw. **13**(2), Article 11:1–24 (2017)
20. M Nikolov, ZJ Haas, Relay placement in wireless networks: Minimizing communication cost. IEEE Trans. Wireless Commun. **15**(5), 3587–3602 (2016)
21. K Ozera, T Oda, D Elmazi, L Barolli, in *International Conference on Broadband and Wireless Computing, Communication and Applications*. Design and implementation of a simulation system based on genetic algorithm for node placement in wireless sensor and actor networks (Springer, Cham, 2016), pp. 673–682
22. R Pallavi, GB Prakash, in *2015 International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*. A review on network partitioning in wireless sensor and actor networks (IEEE, Davangere, 2015), pp. 771–782
23. T Qiu, N Chen, K Li, D Qiao, Z Fu, Heterogeneous ad hoc networks: Architectures, advances and challenges. Ad Hoc Netw. **55**, 143–152 (2017)
24. T Qiu, D Luo, F Xia, N Deonauth, W Si, A Tolba, A greedy model with small world for improving the robustness of heterogeneous internet of things. Comput. Netw. **101**, 127–143 (2016)
25. T Qiu, A Zhao, R Ma, V Chang, F Liu, Z Fu, A task-efficient sink node based on embedded multi-core soc for internet of things. Futur. Gener. Comput. Syst. **12**(24), 1–11 (2016)
26. T Ramezani, T Ramezani, A distributed method to reconstruct connection in wireless sensor networks by using genetic algorithm. Modern Appl. Sci. **10**(6), 50 (2016)
27. V Ranga, M Dave, AK Verma, Network partitioning recovery mechanisms in wsans: a survey. Wirel. Pers. Commun. **72**(2), 857–917 (2013)
28. P Rawat, KD Singh, H Chaouchi, JM Bonnin, Wireless sensor networks: a survey on recent developments and potential synergies. J. Supercomput. **68**(1), 1–48 (2014)
29. I Senturk, K Akkaya, S Janansefat, Towards realistic connectivity restoration in partitioned mobile sensor networks. Int. J. Commun. Syst. **29**(2), 230–250 (2016)
30. L Sitanayah, KN Brown, CJ Sreenan, A fault-tolerant relay placement algorithm for ensuring k vertex-disjoint shortest paths in wireless sensor networks. Ad Hoc Netw. **23**, 145–162 (2014)
31. R Virender, D Mayank, VA Kumar, in *Proceedings of International Conference on ICT for Sustainable Development*. Lost connectivity restoration in partitioned wireless sensor networks (Springer, Singapore, 2016), pp. 89–98
32. G Wang, J Yu, D Yu, H Yu, L Feng, P Liu, Ds-mac: An energy efficient demand sleep mac protocol with low latency for wireless sensor networks. J. Netw. Comput. Appl. **58**, 155–164 (2015)
33. H Wang, X Ding, C Huang, X Wu, Adaptive connectivity restoration from node failure (s) in wireless sensor networks. Sensors. **16**(10), 1487 (2016)
34. C Wu, H Chen, J Liu, in *Consumer Electronics, Communications and Networks (CECNet), 2013 3rd International Conference on*. A survey of connectivity restoration in wireless sensor networks (IEEE, Xianning, 2013), pp. 65–67
35. M Younis, S Lee, AA Abbasi, A localized algorithm for restoring internode connectivity in networks of moveable sensors. IEEE Trans. Comput. **59**(12), 1669–1682 (2010)
36. M Younis, IF Senturk, K Akkaya, S Lee, F Senel, Topology management techniques for tolerating node failures in wireless sensor networks: A survey. Comput. Netw. **58**, 254–283 (2014)
37. J Yu, N Wang, G Wang, D Yu, Connected dominating sets in wireless ad hoc and sensor networks a comprehensive survey. Comput. Commun. **36**(2), 121–134 (2013)
38. X Zheng, Z Cai, J Li, H Gao, A study on application-aware scheduling in wireless networks. IEEE Trans. Mobile Comput. **16**(7), 1781–1801 (2017)