Available online at http://ijim.srbiau.ac.ir/
Int. J. Industrial Mathematics (ISSN 2008-5621)
Vol. 10, No. 3, 2018 Article ID IJIM-01141, 16 pages
Research Article

Connectivity Restoration in Wireless Sensor and Actor Networks using Distributed Learning Automata

M. Jahanshahi *†, M. Maddah ‡

Received Date: 2017-11-09 Revised Date: 2018-03-07 Accepted Date: 2018-05-18

Abstract

In wireless sensor and actor networks, connectivity among actors is very important, especially in critical applications where actors collaborate with each other to provide a report as soon as possible. Sometimes a node failure divides the given network into several parts; causing loss of connectivity between actors hence degrades the network performance. Several algorithms have been proposed to restore inter-actor connectivity. In these methods, selecting appropriate failure handler to minimize the relocations is crucial. In this paper the issue of finding the best backup is supposed the same as finding shortest path in stochastic graph problem. Owing that the solving stochastic shortest path is NP-complete, DLA is used for failure handler choice. This essay also presents a hybrid algorithm named DLA-BuS for critical node Back Up Selection based on distributed learning automata. Here two methods are proposed; first one is that each actor node is equipped with a learning automaton so that their cooperation in learning process leads to select the desired backups. The second method states the presentation of DLA-MRF to repair stimulant failure of two adjacent actors. In order to show the performance of the proposed algorithms extensive simulations using Castalia simulator have been conducted. Simulation results demonstrate that the mentioned proposed algorithms outperform existing methods in terms of the number of nodes movement, total distance travels, the percentage of coverage reduction, and energy consumption.

Keywords : Wireless sensor and actor networks; Fault recovery; Connectivity restoration; Node relocation; Distributed learning automata.

1 Introduction

Recently, the popularity of wireless sensor and actor networks (WSAN) is increasing, since these networks are appropriate for applications such as critical and hard missions requiring intelligent and autonomous interaction with the environment. The wide range of application in WSANs includes important themes such as area, dynamic or static point, and barrier coverage. WSANs are composed of a large number of static sensor nodes and few movable nodes called actor. The sensor nodes senses environment and report events to one or multiple actors for processing, making decisions and performing appropriate actions [1, 2, 3]. Actors communication with

*Corresponding author. mjahanshahi@iauctb.ac.ir, Tel: +(98)9126826209.
†Young Researchers and Elite Club, Central Tehran Branch, Islamic Azad University, Tehran, Iran.
‡Department of Computer Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran.

each other via wireless links. Each node works independently and with no human intervention. Sensors are physically very small and have limitations in processing, memory capacity and power supply. Actors are nodes with more processing power and more powerful battery that should react to the event sensed by the sensor during a specified period time. The role of an actor is extremely crucial for a timely response to events such as fire, earthquakes, disasters, etc., and depends on the environment and capabilities of actors that may vary from one application to another [4, 5, 6].For example, an actor can extinguish a fire, lift rubbles, rescue trapped survivors, deactivate a landmine and carry weapons.

The most problem of WSANs in harsh environments is network partitioning due to the failure of one or more critical actor nodes [7].For recovering such partitions, sometimes additional mobile nodes can be used such as unmanned aerial vehicles (UAVs) or robots. However, these relay nodes are expensive and difficult to deploy due to unattended environment, where exact locations are unknown. A cost effectiveness and real time solution is moving the implicit suitable actor nodes for recover network partitioning in WSANs [8, 9]. There are three methodologies to recover a partitioned network: proactive, reactive and hybrid. Proactive approaches establish and maintain bi-connected topology in order to provide fault tolerance capability. These techniques require large communication overhead and are not cost effective. On the other hand, reactive approaches start recovery procedure once an actor node detects the failure. Therefore, these approaches are not suitable for time sensitive applications. The third methodology is hybrid approach that is comprised of proactive and reactive phases. In proactive phase, local information within k-hop neighborhood is stored in each node for the recovery. In reactive phase, instantaneous action is performed using the stored information in case that partitioning occurs.

In this paper, two algorithms named DLA-BuS and DLA-MRF are proposed both for connectivity restoration. These approaches use distributed learning automata to designate suitable backup for critical actors. DLA-BuS recovers the network in which one actor is failed at each time, while DLA-MRF is proposed to handle failures of multiple actors.

In this method, we prioritize neighbors of the critical actor. Like DLA-BuS, the recovery procedure is applied iteratively until required connectivity is restored. Simulation results demonstrate the performance of the proposed approaches in terms of total distance movement, nodes involved in recovery, message passing and percentage of coverage reduction.

The rest of the paper is organized as follows: Section 2 reviews the related works.section 3 describes the concept of learning automata, distributed learning automata, and also variable action-set learning automata theories. In Sections 4 and 5, the DLA-BuS and DLA-MRF are discussed, respectively. Performance of the proposed algorithms is evaluated through the extensive simulation experiments in Section 6. Finally Section 7 concludes the paper.

2 Related works

The fault tolerance issue in WSAN has been considered in few works in different contexts. In [10] fault-tolerance is achieved by means of redundancy. In other words, sensors send their sensed data to more than one actor and each actor receives the sensed information from multiple sensors in the event area. In this paper, we focus on network partitioning caused by node failure rather than reliable event notification delivery. In this paper, we briefly review some of the existing connectivity restoration algorithms for WSAN. Some approaches use proactive strategy to prevent network from partitioning. For example,[11] provides k-connectivity. This idea is referred to as k-vertex connectivity where the failure of (k -1) nodes does not lead to any network partitioning problem. The aim of these approaches is to form k-node disjoint paths between pairs of nodes in the network. Another distributed approach proposed in [12] to restore 2-connectivity. Unlike these approaches, our algorithm focuses on providing 1-connectivity. Such an optimization is a very challenging problem that has been proven

to be NP-hard [13] Therefore, prevention strategies have tolerated high message overhead.
Some researchers employed node mobility for network portioning recovery. The approaches that attempt to prevent repartitioning network, caused by node mobility, are categorized into block and cascaded movement. In block movement, all of nodes within a block, are moved to recover network partitioning. Block movement, as defined in [14, 15], is a solution based on the relocation of all nodes within a partition (block). Specifically, in each partition a node will be leader and moves. The neighbor nodes of a leader node, follow it to recover the dis-connectivity. Instead of using block movement, DLA-BuS pursues a cascaded relocation. The idea is to gradually replace intermediate nodes on the path instead of moving a node for a long distance.
VCR [16] and PADRA [17] are proactive methods and use 2-hop neighbors information. PADRA tries to find a connected dominating set that increases message passing overhead. PADRA uses a greedy algorithm to designate appropriate backup for the failed node. The ideal failure handler will be a dominatee neighbor of Ai, which can be simply replaced by Ai. If a dominatee is not available, the closest dominator is candidate as the failure handler of Ai. Repositioning the failure handler continues until a dominatee is encountered. Greedy algorithms of these approaches may not lead to accurate solution.
Another solution to recover a partitioned network was proposed in [18]. DARA is a reactive approach that like PADRA uses 2-hop neighbor information. The idea is to replace the dead actor node by a suitable candidate as failure handler based on nodes degree and distance. The overall objective of DARA is to minimize the total distance travelled by the involved actors in order to limit the movement overhead. 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 the whole topology. The selection of failure handler (FH) to replace the failed node is done based on the neighbors degree and distance to failed node which may require excessive replacement until a leaf node is found. Sometimes, moving distance may be long and will consume more energy of the network; also the best candidate selection is done reactively using a lot of calculations, which is very crucial for delay sensitive applications.
Younis et al. [19] suggested a localized reactive distributed algorithm called recovery through inward motion (RIM) for the network partition recovery. The main idea is to relocate the whole neighbor of a failed node Af towards the position of Af so that they would be able to reach each other. RIM likes that our proposed DLA-BuS reduces the message overhead by maintaining only 1-hop neighbor information but a large number of relocation are required for the recovery. Also, it does not differentiate between critical and non-critical nodes.
In [20], authors presented Coverage aware Connectivity Restoration algorithm (C3R). C3R is a reactive method that uses 1-hop neighbors of failed node to restore connectivity. When neighbors detect failure of a node, take turns in moving to the position of F. After serving for some time, each neighbor goes back to its original position, allowing for another neighbor of F to come forward and so on. In C3R the coverage stays mostly the same as its pre-failure status. Another approach has been proposed to improve C3R in [21]. The aim of DRFN (Detection and Replacement of a Failing Node) is that the consumed total energy, for the restoration of connection, would be shared by neighbor nodes extending the whole network lifetime. DRFN ranks the nodes according to the energy, distance and node's degree. These methods consider coverage but neighbors will consume more energy with more chances to fail.
In some research works, authors proposed hybrid approaches like DCR [22] and EDCR [23]. DCR distributes partitioning detection and connectivity restoration algorithm and EDCR is energy aware DCR. DCR proactively identifies actors that are critical to the network connectivity (primary), and designates appropriate backup nodes. Upon failure detection, the backup actor initiates a recovery process that may involve coordinated relocation of multiple actors. DCR uses 1-hop neighbors information to repair connectivity while imposing minimal communication overhead. EDCR has been proposed to decrease the number of heartbeat messages in DCR. In the reactive part, once nodes energy falls below a cer-

tain threshold it informs its one hop neighbors. Neighbor nodes start monitoring of the primary through HEARTBEATS. Once neighbor nodes detect the failure they buer critical data to prevent from losing sensed data. The backup node is replaced by the primary and it informs neighbors resuming transmission of their buered data packets. These ideas are similar to that of PADRA but no CDS is employed which degrades the accuracy of assigning appropriate backups and also convergence of greedy algorithms are not mentioned. Also, EDCR does not consider physical failures and energy as two factors in designating backup node.

CoMN2 was proposed to repair the partitioned network from multi adjacent failures. However, it is centralized and considers the presence of specific entity called mobile node which has the capability of processing data and making the appropriate decision. In this approach, mobile node is a single point of failure [24]. Another idea for recovery the partitioned network from multi node failure is RAM [22]. RAM like DLA-MRF is a hybrid approach but each backup node should be aware of its grandparent and like DCR does not consider energy of node for selecting appropriate backup.

In this paper we present DLA-BuS approach that like RIM only requires that each node knows the locations of its 1-hop neighbors. In addition, like DCR, it uses a localized algorithm to identify critical actors and here, each critical node has only one backup to handle its failure.

3 Learning automata, distributed learning automata and variable action-set learning automata

In this section, we briey review learning automata, distributed learning automata and variable action-set learning automata.

3.1 Learning automata theory

A learning automaton [25, 26, 27, 28, 29, 30, 31, 32] is an adaptive decision making unit that improves its performance by learning how to choose the optimal action from a finite set of allowed actions through repeated interactions with a random environment. The action is chosen at random based on a probability distribution kept over the action-set and at each instant the given action is served as the input to the random environment. The environment responds to the taken action in turn with a reinforcement signal. The action probability vector is updated based on the reinforcement feedback from the environment. The objective of a learning automaton is to find the optimal action from the action-set so that the average penalty received from the environment is minimized.

The environment can be described by a triple $\{\alpha, \beta, c\}$ where $\alpha \equiv \{\alpha_1, \alpha_2, .., \alpha_r\}$ represents the finite set of inputs (actions) $\beta \equiv \{\beta_1, \beta_2, \ldots, i_m\}$ denotes the set of values can be taken by the reinforcement signal, and and c=$\{f(\alpha) \mid \alpha \epsilon \alpha\}$ denotes the set of the penalty irobabilities (probability distributions over β) $f(\alpha)$ is called penalty function. If the penalty probabilities are constant, the random environment is said to be a stationary random environment and if they are time-variant, the environment is called a non - stationary environment. The environments depending on the nature of the reinforcement signal b, can be classified into P-model, Q-model and S-model. The environments, in which the reinforcement signal can only take two binary values 0 and 1, are referred to as P-model environments. Another class of the environment allows a finite number of the values in the interval [0,1] can be taken by the reinforcement signal. Such an environment is referred to as Q-model environment. In S-model environments, the reinforcement signal lies in the interval [0,1]. The relationship between the learning automaton and its random environment has been shown in Fig. 1. Learning automata can be classified into two main families [33, 34, 35]:fixed structure learning automata and variable structure learning automata. Variable structure learning automata are represented by a triple $< \underline{\beta}, \underline{\alpha}, T >$, where β is the set of inputs, α is the set of actions, and T is learning algorithm. The learning algorithm is a recurrence relation which is used to modify the action probability vector. Let $\alpha(k)$ and $\underline{p}(k)$ enote the action chosen at instant k and the action probability vector

on which the chosen action is based, respectively. The recurrence equation shown by (3.1) and (3.2) is a linear learning algorithm by which the action probability vector p is updated. Let $\alpha_i(k)$ be the action chosen by the automaton at instant k. The action probabilities are updated as given in Eq. (3.1),when the chosen action is rewarded by the environment (i.e., β(n)=0). When the taken action is penalized by the environment, the action probabilities are updated as defined in Eq. (3.2) (i.e., β(n)=1).

$$\begin{aligned} p_i(n+1) &= p_i(n) + a[1 - p_i(n)] \\ p_j(n+1) &= (1-a)p_j(n) \forall j, j \neq i \end{aligned} \tag{3.1}$$

$$\begin{aligned} p_i(n+1) &= (1-b)p_i(n) \\ p_j(n+1) &= \frac{b}{r-1} + (1-b)p_j(n) \forall j, j \neq i \end{aligned} \tag{3.2}$$

Where, r is the number of actions can be chosen by the automaton, a and b denote the reward and penalty parameters and determine the amount of increases and decreases of the action probabilities, respectively. If $a = b$, learning algorithm is called linear reward-penalty (L_{R-P})). If $a >> b$ the given algorithm is called linear reward-ϵ penalty ($L_{R-\epsilon P}$), and finalay if $b = 0$ it is called linear reward-Inaction (L_{R-I}). In the latter case, the action probability vectors remain unchanged when the taken action is penalized by the environment.

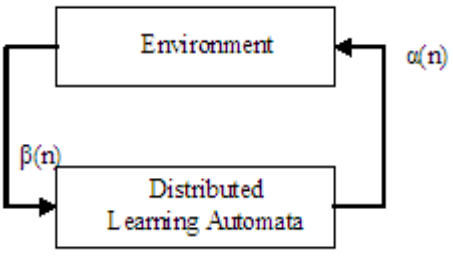

Figure 1: The relationship between the learning automaton and its random envirohment.

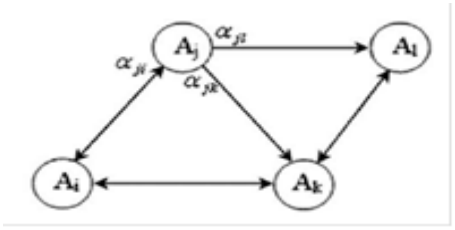

Figure 2: Distributed learning automata.

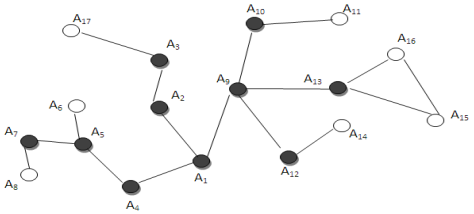

Figure 3: An example of a connected topology.

3.2 Distributed learning automata

A distributed learning automata (DLA) [36] is a network of the learning automata which collectively cooperate to solve a particular problem. Formally, a DLA can be defined by a quadruple $< A, E, T, A_0 >$, where $A = \{A_1, A_{2,...,A_m}\}$ is the set of the learning automata,$E \subset A \times A$ is the set of the edges in which edge $e_{(i,j)}$ corresponds to the action α_j of the automaton A_i, T is the set of learning schemes by which the learning automata update their action probability vectors, and A_0 is the root automaton of DLA from which the automaton activation is started. An example of a DLA has been shown in Fig. 2. The operation of a DLA can be described as follows: At first, the root automaton randomly chooses one of its outgoing edges (actions) according to its action probabilities and activates the learning automaton at the other end of the selected edge. The activated automaton also randomly selects an action which results in activation of another automaton. The process of choosing the actions and activating the automata continues until a leaf automaton (an automaton which interacts with the environment) is reached. The chosen actions, along the path induced by the activated automata between the root and leaf, are applied to the random environment. The environment evaluates the applied actions and emits a reinforcement signal to the DLA. The activated learning automata along the chosen path update their

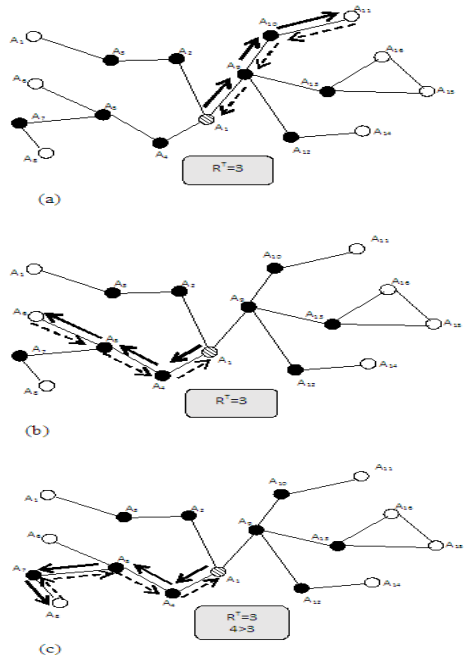

Figure 4: An example of selection backup whit DLA.

action probability vectors on the basis of the reinforcement signal by using the learning schemes. The paths from the unique root automaton to one of the leaf automata are selected until the probability, by which one of the paths is chosen, is close enough to unity. Each DLA has exactly one root automaton which is always activated, and at least one leaf automaton which is activated probabilistically.

3.3 Variable action-set learning automata

A variable action-set learning automaton is an automaton in which the number of actions available at each instant changes with time. It has been shown in [31] that a learning automaton with a changing number of actions is absolutely expedient and also e-optimal. Such an automaton has a finite set of n actions,$a = \{a_1, a_2, ..., a_n\}, z = \{z_1, z_2, ..., z_m\}$ denotes the set of action subsets and is the subset of all the actions can be chosen by the learning automaton, at each instant k. The selection of the particular action subsets is randomly made by an external agency according to the probability distribution $q(k) = \{q_1(k), q_2(k), ..., q_m(k)\}$ defined over the possible

DLA-BuS(){
1. ET=0
2. RT= n-1
3. If (IsCritical==true)
4. Assign backup with DLA()
5. If A detect the failure of primary then
6. Notify to backup
7. Recovery(A,F)

}
Assign backup with DLA (){
8. Form action set(){
9. for(i=0; i< NeighborTable.size() ; i++)
10. for(j=0; j< NeighborTable.size() ; j++)
11. If(NeighborTable[i] != TrN-ID[j])
12. ActionSet.pushback(NeighborTable[i])
13. }
14. Choose a neighbor randomly()
15. selection:next = rand()16. if(action[next].Active==true)
17. ActiveNeighbor = action[next].Id
18. else
19. goto selection;
20. }
21. Put Node ID in TrN-IDarray and send ACTIVATION message (vector TrN-IDarray){
22. TrN-ID-array. pushback (NodeID ())
23. for(int i=0; i¡(int) TrN-IDarray.size(); i++)
24. ACtivationPacket ⟶ setNodeID (i, TrN-IDarray[i]) //create Activation packet
25. SendACTIVIATIONmessage(ACtivationPacket, ActiveNeighbor);
26. }
27. If Hi received ACTIVATION message{
28. if (IsCritical!=true | (int) NeighborTable.size()==1 | actionSet.size()==0){
29. Calculatewieght()
30. count=1
31. SendResponsemessage(ResponsePacket, nodeWhichReceivedActivationMessageFromIt)
32.

33. else if (IsCritical==true)
34. goto 11
35. }

36. If Hi received RESPONSE message{ 37. if (count ≤ RT){
38. RT = count
39. Reward Action(aR) //reward action by replacement threshold
40. }
41. else if (*count* > *RT*){
42. Penalize Action (bR) //penalize action by replacement threshold
43. }
44. if (*weigh*/*count*) ≤ *ET* {
45. *ET* = (*weigh*/*count*) 46. Reward Action(aE) //reward action by energy threshold
47. }
48. else if (*weigh*/*count*) ≥ ET {
49. Penalize Action(bE) //penalize action by energy threshold
50. }
51. count++
52. Calculatedweigh();
53. Weight=weightHi+ weightrecieved
54. SendResponsemessage(ResponsePacket, nodeWhichReceivedActivationMessageFromIt)
}

Recovery (A, F){
55. Move to location F (F is failed node)
}
Calculatewieght () {
56. RemainingEnergy = ((initEnergy - spentEnergy) * 100)/initEnergy;
57. weightHi = (remainingEnergy / (alfa * (int)neighborTable.size()));
}
Reward Action (a){
58. for(int i=0; i<(int)action.size(); i++){
59. if(action[i].Id == NodeID)
60. action[i].prob[stage+1]=((action[i].prob[stage])+(a*(1-action[i].prob[stage])))
61. else if(action[i].Id != primary and action[i].Active==true)
62. action[i].prob[stage+1] = ((1-a)* action[i].prob[stage+1];
63. else if (action[i].Active==false)
64. action[i].prob[stage+1] = action[i].prob[stage];
65. }
}
textbfPenalize Action (b) {
66. for(int i=0; i¡(int)action.size(); i++){
67. if(action[i].Id == NodeID)
68. action[i].prob[stage+1]= ((1-b)*action[i].prob[stage])
69. else if(action[i].Id != NodeID and action[i].Active==true){
71. r = (int)action.size()-1;
72. action[i].prob[stage+1]=((*b*/(*r* − 1))+action[i].prob[stage]*(1-b));
73. }
74. else if (action[i].Active==false)
75. action[i].prob[stage+1] = action[i].prob[stage];
76. }
}

Figure 5: Pseudo code of DLA-BuS.

subsets of the actions, where $q_i(k) = prob[z(k) = Z_i \mid Z_i \in Z, 1 \leq i \leq 2^n - 1]$.$p_i(k) = prob[a(k) = a_i \mid Z(k), a_i \in Z(k)]$ is the probability of choosing action a_i, conditioned in the event that the action subset has already been selected and also $a_i \in Z(k)$.The scaled probability is defined as fallow:

$$\widehat{P}_i(k) = \frac{p_i(k)}{R(k)} \tag{3.3}$$

Where,$R(k) = \sigma_{a_i \in Z(k)} p_i(k)$ is the sum of the probabilities of the actions in subset . The procedure of choosing an action and updating the action probabilities in a variable action-set learning automaton can be described as follows. Let $Z(k)$ be the action subset selected at instant k. Before choosing an action, the probabilities of all the actions in the selected subset are scaled as defined in Eq. (3.3). The automaton then randomly selects one of its possible actions according to the scaled action probability vector $\widehat{P}_i(k)$.Depending on the response received from the environment, the learning automaton updates its scaled action probability vector. Note that the probability of the available actions is only updated. Finally, the probability vector of the actions of the chosen subset is rescaled as:

$$P_i(k+1) = \widehat{P}_i(k+1).R(k) \forall \alpha_i \in Z(k) \tag{3.4}$$

The absolute expediency and e-optimality of the method described above have been proved in [28]. Variable action set learning automata have been found to perform well for solving combinatorial optimization problems.

4 DLA-based BackUp Selection

The proposed method which is called Distributed Learning Automata-based BackUp Selection (DLA-BuS) is a hybrid approach to recover partitioned network. All nodes store a list of their one hop neighbor nodes information and exchange heartbeat messages to update the neighborhood table. In fact, we are seeking to answer the following two questions: 1. Which one-hop neighbor of node Ai (the faulty node) is more appropriate for re-location?
2. How should we prevent the network from repartitioning?

In order to answer the first question, we use distributed learning automata to select a candidate node with more energy and less relocation as the failure handler. Cascade relocation is used in order to answer the second question. Finding a desirable solution requires each node to be aware of the whole network topology information. These centralized methods are not applicable and scalable in practice. We aim to propose a localized and distributed method using exploitation of learning algorithms to find a sub optimal solution.

4.1 Problem statement and Assumptions

Topology of the given network is assumed flat. Nodes are uniformly deployed. It is assumed that nodes are aware of their location through GPS or other localization algorithms. Nodes communication range is denoted by r_c . The sensing range and communication range of all nodes are assumed to be identical.
As mentioned earlier, hybrid approaches are more appropriate for time sensitive applications. In this paper, we proposed a hybrid approach that includes a reactive and a proactive phase. In the proactive phase, critical nodes are identified by a localized method and then, the most appropriate node is selected as backup from 1-hop neighbors using distributed learning automata. In the reaction phase, our proposed approach uses cascade relocation to prevent from repartitioning.

4.2 Identification of critical nodes

As described earlier, the failure of critical nodes divides the network into disjoint segments. Our goal is to identify a backup for each of these critical nodes. Various algorithms have been presented for identification of critical nodes. These methods are classified into centralized and distributed categories. In centralized algorithms, each node should be aware of the whole network topology. Because of dynamic nature of networks, these methods impose high communication overhead. Therefore, designing localized and distributed algorithms is a must. Most distributed algorithms require 2-hop neighbors infor-

mation. Similar to DCR, DLA-BuS uses a local method to detect cut vertexes. This distributed method only requires 1-hop neighbors information [37]. In this method, each node determines whether it is critical or not, using the location of neighbors by calculating the distance between each pair of the neighbors. If this distance is less than their communication range, it is known as a non-critical node because the neighbors will still be connected in its absence; otherwise, the node is critical.

Figure 3 shows an example of network topology. Critical and none-critical nodes are indicated with black and white circles, respectively. As it can be seen, A_{16} is 1-hop none critical node because its neighbors remains connected without it but, failure of A_1 divides network into two sub graphs.

4.3 Selection of appropriate backup nodes

In this phase, each critical node should select a neighbor as backup from its 1-hop neighbors. The appropriate backup is a node that leads to minimum relocation and consequently, minimum changes in the network topology. In fact, we are looking for the shortest path to a non-critical node or a leaf node [[38],[39],[40],[41]], and since it is proved that finding the shortest path is an NP-complete issue, distributed learning automata is used to cope with this problem [42]. Here, WSAN is mapped to a two-dimensional weighted graph whose vertices correspond to the actor nodes. The weight of the actor is calculated by Eq. 4.5. Two actors that are located in the communication range of each other are assumed to be connected. Factors α , β are tuned experimentally. Parameter *distance* specifies the distance between a node and its neighbors and also, *Energy%* is percentage of residual on-board energy of the node.

$$weight_i = \frac{Energy\%}{\alpha \times numberOfNgb + \beta \times dist} \quad (4.5)$$

4.3.1 Modeling of environment, action set of learning automata, and identification messages

In this paper, automatas environment is assumed as a network that actors are distributed randomly. The environment is defined as triple $E = \{\alpha, \beta, c\}$. Distributed learning automata contain a network of automata cooperate with each other to find the desirable response. In DLA-BuS, each critical node is equipped with a learning automaton and so a network of learning automata, isomorphic to the network graph, is formed and nodes action-set includes its 1-hop neighbors. The resulting network of learning automata can be described by a duple $< \underline{A}, \underline{\alpha} >$, where $\underline{A} = \{A_1, A_2, ..., A_m\}$ denotes the set of the learning automata corresponding to the vertex set, and $\underline{\alpha} = \{\alpha_1, \alpha_2, ..., \alpha_m\}$ denotes the set of actions in which $\alpha_i = \{\alpha_{i1}, \alpha_{i2}, ..., \alpha_{im}\}$defines the action-set can be taken by learning automata A_i.

In DLA-BuS, each critical node forms its action-set by its neighborhood table. To prevent from the loop and increasing convergence speed, we apply the learning automata with variable action-set, and introduce the following rule for pruning the action-set of such learning automata. Rule 1: Each node has a dynamic array called Transmitter Nodes ID that simplified as TrN-ID array. TrN-ID array includes nodes ID which ACTIVIATION message received through them. In each iteration, activated learning automaton is allowed to prune its action-set by disabling the actions corresponding to the 1-hop neighbors that their IDs exist in TrN-ID array.

In DLA-BuS we assume two types of messages; ACTIVATION message and RESPONSE message. ACTIVATION message includes a dynamic message of the transmitter nodes' identifiers. When a given host H_i receives an ACTIVATION message, it activates its corresponding learning automaton and inserts its own ID into the TrN-ID array. The second message is RESPONSE message. It includes two fields, i.e., weight and count which are used to reward and punish the chosen action.

4.3.2 Selection of backup node using P-model

Each node A_i is equipped with a learning automaton (LA). As discussed in section 4.2, once H_i is detected as critical node activates its corresponding automaton (LA_i) and then, forms its action-set according to its probability vector. At first

iteration, all of actions have the same probability and then the probability vector changes over time by rewards and punishments they receive from environment. The node that starts backup selection process (A_i) is known as the root node. Since all of actions have the same probability at first, so A_i selects one of its actions randomly, pushes its node ID to TrN-ID array and sends ACTIVATION message to corresponding neighbor.

An ACTIVATION message includes TrN-ID array that described in section 4.3.1. When a node A_i receives an ACTIVATION message, if it is critical, activates its corresponding LA and updates its action-set by TrN-ID array. Then it selects one neighbor and sends activation message to it. If A_i receives activation message and it is a leaf or non-critical node, creates a RESPONSE message with following values. A_i calculates its weight by Eq. (4.5) and sets count filed value as 1. Then, A_i sends this information to the node that was sender of the activation message. If A_i is a none-root node and receives RESPONSE message, compares count value with the dynamic relocation threshold (R^T). R^T contains the minimum nodes relocation which has been required yet in case of failure root node. This is initially set to the network size. If the count is less than or equal to the relocation threshold, action α_{ij} rewarded and probability of it (p_{ij}) increased by Eq. (3.1). Otherwise, it is punished and its probability decreases by Eq. (3.2). After a node updates its probability vector, it produces a new response message. In this case, the node increments the value of count then calculates its weight and adds it to the weight which received from response message. Afterwards sends the response message to the actor, which was the sender of the activation message.

$$weight_i = \frac{Energy\%}{\alpha \times numberOfNgb + \beta \times dist} \quad (4.6)$$

In DLA-BuS, two measures are used for reward and punishment. First, relocation which was explained above and next average energy. The purpose is selecting a node as a backup that imposes low cascade relocation to the network and have a higher energy level to increase the lifetime of the network. In order to evaluate average consumed energy, we define an energy threshold (E^T). E^T contains the average residual energy of nodes which has been needed for nodes relocation in case of failure root node. E^T is initially set at zero. Once root node receives response message, it calculates $weight_{avg}$ by Eq. (4.6). If is greater than or equal to the energy threshold (E^T), action α_i is rewarded and its probability is increased. Otherwise, it is punished and its probability is decreased. When A_i receives RESPONSE message and it is root node, then updates its vector probability as described above and then selects a neighbor with the highest probability as a temporary backup node. The stopping condition of the Backup selection process is when the probability of one action becomes greater than or equal to . is empirical variable and will be determined in section 6. The flowchart and pseudo-code of the suggested method are shown in Fig. 5 and Fig. 6.

4.3.3 Selection of backup node using S-model

In section 4.3.2 we proposed our idea of learning automata only based on the P-model environment, where the input to the automaton is either 0 or 1. Now, in this section S-model is used that considers the possibility of input values over an interval $[LB, UB]$ and the environment response is given by Eq. (4.7). Since in S-models the outputs lie in the interval [0,1], and therefore are neither totally favorable nor totally unfavorable, the main problem is to determine how the probabilities of all actions are to be updated. In the case, the environment outputs $\beta_i(n)$ are not in the interval [0,1], but in $[LB, UB]$, it is always possible to map the output into the unit interval using Eq. (4.8):

$$\beta_i = \frac{\Sigma_{k=1}^{count} weight_{count}}{count} \quad (4.7)$$

$$\beta_i = \frac{\beta_i - a}{LB - UB} \quad (4.8)$$

$$LB_i = \min\left(\frac{Energy\%}{\alpha \times numberOfNgb + \beta \times dist}\right) \approx 0 \quad (4.9)$$

$$UB_i = \min\left(\frac{Energy\%}{\alpha \times numberOfNgb + \beta \times dist}\right) \approx 100 \quad (4.10)$$

According to Eq. (4.9) and (4.10) $LB = min\{\beta_i\}$ and $UB = max\{\beta_i\}$. Therefore, only the normalized S-model will be considered. The general procedure is similar to the P-model.

4.4 Failure detection and recovery

In DLA-BuS, nodes use heartbeat messages to monitor critical nodes and update its neighborhood table. Each neighbor node exchanges heartbeat message periodically with its primary (critical node). When a backup node does not receive any heartbeat message from a critical node, it is assumed that critical node is failed and then recovery process is executed.

4.5 Recovery process

In case a Backup node detects the failure of its critical node, this procedure begins. Thus, backup node moves to location of failed node and repair partitioned network. DLA-BuS uses cascade movements to avoid repartitioning network. Fig . 4 shows an example of repair process.

5 DLA-based Recovery partitioned network from Multiple Failure

As mentioned earlier, the goal of DLA-BAS is to recover partitioned network from single node failure. However, some events may cause the failure of more than one adjacent actor. In general, the recovery from simultaneous failure of multiple nodes is very challenging. DLA-BuS and single failure recovery methods are not guaranteed to converge. For example, consider the topology of Fig. 4. In Fig. 7(a), A_1 is selected as a backup for node A_2 and A_3. Once A_2 and A_3 fail simultaneously and A_1 detects failure of them, it moves to repair the one that is detected first. Therefore, none of the surviving nodes is responsible for tolerating the failure of the other critical failed node and network not recovered. Also, Fig. 7(b) shows the scenario that critical node and its backup are failed simultaneously. In this section, we propose an algorithm to improve DLA-BuS for Recovery partitioned network from Multiple Failure used Distributed learning Automata (DLA-MRF). DLA-MRF like DLA-BuS is a hybrid approach, but the criteria for backup selection and recovery are different. The details of the backup selection and recovery procedure are described later in this section.

5.1 Backup Selection

DLA-MRF uses distributed learning automata for backup selection the same as DLA-BuS that described in section 4. However, it has more than one backup for each critical node. Once a node is identified as critical, the LA which is resident on the node, is activated. In each iteration of learning process, neighbors are prioritized from 1 to k according to automatas action probability vector. The neighbor node associated with the most probability is selected as backup. If two nodes have the same probability, the node with bigger ID will have higher priority.

5.2 Failure detection and recovery

In DLA-MRF, neighbors exchange heartbeat messages to monitor critical nodes. In our proposed scheme when any heartbeat message is received from S_i then, S_i neighbor nodes assume it is failed and initiates the recovery procedure. Here, backup node moves toward the location of the failed node. Once it reaches the location, broadcast DONE RECOVERY message to inform all neighboring nodes from accomplished recovery process. DLA-MRF uses a timer to prevent neighbors from simultaneous recovery process executions. Actually, after failure detection each neighbor node checks its priority and waits $T \times (k-1)$ seconds (k is nodes priority). If DONE RECOVERY message is not received after waiting time, then that node moves toward the location of the failed node. T is calculated by Eq. (5.11).

$$T = 2\frac{L_{HB}}{R} + \frac{d_{max}}{s} \tag{5.11}$$

According to Eq. (5.11) L_{HB} is the length of Heartbeat packets in bits, $\boldsymbol{R}$ is heartbeat packet rate in bit per second, d_{max} is the longest distance between two neighboring nodes, and $\boldsymbol{s}$ is movement speed of mobile node. In other word,

$\boldsymbol{T}$ is proportional to round-trip time of a heartbeat packet (time of failure detection) and time taken by the actor for moving towards location of failed node (moving time). Transmission delay and propagation delay is assumed to be negligible. Fig. 8 depicts the flowchart of DLA-MRF algorithm.

6 Performance analysis of DLA-BuS and DLA-MRF

In two next sub-sections, we will present simulation setup and performance metrics of DLA-BuS and DLA-MRF, respectively.

6.1 Experiment 1: Simulation setup and performance metrics of DLA-BuS

Inter-actor topologies consist of a varying number of nodes are considered in this set of experiments. The goal of the simulations is to study the performance of the purposed approach compared to DCR, RIM, PADRA, and DARA. The comparisons are evaluated on Castalia 3.2, a simulator for wireless sensor networks that is based on the OMNeT++ platform. Castalia allows to realistically modeling the sensor nodes as well as wireless channel. The radio device is accurately modeled having different radio states with individual energy consumptions and transition times. The wireless channel uses an advanced path loss model that is based on experiments and calculates the packet probability using a signal to interference-plus-noise ratio (SINR) model. As radio model the TI CC2420 transceiver is used. It is designed for low-power and low-voltage wireless applications. The energy consumption for receiving and idle listening is 62 mW, whereas the transmission cost is 57 mW (0 dBm). For duty cycling the radio is put into the power down mode, decreasing the power consumption to 0.072 mW. The initial energy is set at 18720 joule. Also, a simple CSMA/CA Mac protocol is used. Nodes are randomly deployed in an area of 10001000 square meter with no obstacles that hinders a node from moving to a new position. We have varied the transmission range of actors between 50 and 200 m so that the topology becomes strongly connected. In what follows the outcome of simulation results conducted for evaluation of methods are presented:

Number of nodes moved: this metric shows the number of nodes that are involved in recovery procedure. Number of moving nodes is a function of energy consumption in the network (i.e. network lifetime). More energy will be consumed, if more nodes move in the network. Fig.9 shows the number of moving nodes in DLA-BuS in comparison with the four algorithms DCR, DARA, RIM and PADRA. This is because in our proposed approach learning automata learn which neighbor causes the least relocation in the network and limits the region of recovery after selecting the nearby none-critical node as backup. In contrast to DLA-BuS in which just the backup node moves to restore the connectivity, in reactive methods such as RIM and DARA all neighboring nodes participate in the recovery process and hence, the number of moving nodes increases. Furthermore, the performance of DLA-BuS remains almost constant with varying the number of the nodes, which implies scalability in the large scale networks. Fig. 9(b) captures the impact of the node movement as a function of nodes radio range variations for the network of 50 nodes. As nodes radio range increases, the number of traveling nodes decreases and becomes almost constant in all approaches as it is expected. However, DLA-BuS further reduces the number of the participated nodes for the recovery as explained earlier.

Total distance moved: This metric demonstrates the total distance travelled by involved nodes until the connectivity is restored. Fig. 10 shows the distance traveled by all nodes in DLA-BuS via DCR, RIM, PADRA and DARA. As observed, nodes traveled less distance than the other approaches in DLA-BuS. Total distance is directly related to the number of nodes moved. As both Figs indicate, performance of DLA-BuS remains almost constant even with higher network densities and longer transmission ranges. *Message passing overhead*: this metric reports the messaging overhead as a function of the network size and nodes radio ranges. fig. 11 shows message passing overhead in DLA-BuS. As it can be observed in the figure, the resultant overhead of DLA-BuS is less than the methods which use 2-

hop neighbors information such as PADRA and DARA. However, DLA-BuS leads to more massaging overhead compared to RIM and DCR due to exchange of control messages among LAs. In other words, in connectivity restoration methods, nodes have to exchange their neighborhood tables with each other. Therefore, in some approaches like PADRA and DARA that use 2-hop neighborhood information, nodes have to exchange too much volume of data in order to select backup node.

Coverage reduction: Another metric to evaluate the connectivity restoration ideas is the percentage of coverage reduction with respect to prior of failure. This metric assesses the effectiveness of the proposed approaches in terms of mitigating the coverage loss. Although, connectivity is the main objective of these approaches, the network coverage is also vital for many real-time applications. It is calculated regarding the changes in nodes position (area coverage) as compared to the original positions. Figure 12 shows the reduction of coverage relative to pre-failure level as the function of number of involved nodes in the recovery process. Again, DLA-BuS provides better results over DARA, PADRA, RIM and DCR due to small changes in the position of the nodes by selecting nearby backup.

Energy Consumption: Fig. 13 demonstrates the total energy consumption in the given network as a function of network density and nodes radio ranges. In connectivity restoration algorithms, energy consumption of nodes is affected by two factors: first, movement overhead and the second is message passing overhead. Movement overhead is directly related to the number of cascade relocations. In other words, nodes consume considerable energy to move from one point to another. From Fig. 13, it is obvious that DLA-BuS compared to other approaches consumes less energy once the network size is increased. Although, DLA-BuS leads to more messaging overhead than DCR and RIM, regarding less relocations (see Fig. 9) DLA-BuS decreases total investigated energy than DCR, RIM, DARA and PADRA and hence, increases the network life time.

6.2 Experiment 2: simulation setup and performance metrics of DLA-MRF

We use Castalia simulator and the same simulation setup to evaluate the performance of DLA-MRF. Transmission delay and propagation delay are assumed to be negligible. Other simulation parameters are shown in fig 14. For more accuracy, we run codes 10 times with 10 different topologies. Each time after identifying critical actors, we choose two adjacent cut-vertices at random to be failed simultaneously.

Fig. 15 (a) shows a comparison of message passing overhead in RAM and DLA. Similarly, Fig. 15 (b) shows a comparison of energy consumption in RAM and DLA via number of nodes. As it can be seen it is obvious that DLA-MRF has more residual energy than RAM due to the selection of the nearby suitable neighbor node with balanced energy consumption in the network during its movement. Also, because of distributed manner, DLA-MRF is sutible for large scale network and its energy consumption results has less variasions than RAM. Although, DLA-MRF uses distributed learning automata with more message passing overhead, it is noticible that it has less relocation and momvent overhead reducing the consumption energy.

Increasing the network density causes that nodess degree increases and hence, the number of critical nodes reduces. Therfore, the total number of replacements and total distance traveled by nodes are decreased. In other words, the availability of non-critical nodes reduces the region of cascaded relocations. Fig. 16 and 17 show results of employing DLA-MRF. DLA-MRF has better performance than RAM for two following reasons:

1. DLA-MRF utilizes distributed learning automata with more accuracy to choose appropriate backup.
2. In RAM a backup node must be aware of its grand primary node.

For example, if A is selected as backup of B and B is backup of C, A should be aware of C to recover network in case of simultaneously failures of B and C. In this idea, A should repair failure of two nodes and hence, A consumes more energy decreasing network lifetime. In contrast, DLA-MRF prioritizes neighbors and uses all neigh-

bor nodes to repair the partitioned network and hence, it provides uniform energy consumption in the network increasesing its lifetime.

7 Conclusion

In this paper, a distributed hybrid approach using Distributed Learning Automata was proposed to restore connectivity of a partitioned network. DLA-BuS selects a suitable neighbor as backup for critical node, and then utilizes cascade relocations during the recovery process instead of moving a complete block of nodes. Afterwards, we presented DLA-MRF for recovering the network from failure of two adjacent nodes. The performance of DLA-BuS and DLA-MRF were validated through extensive Castalia simulations. Simulation results have confirmed the performance of our approaches in term of movement overhead and consumed energy. Also, DLA-MRF balances the energy consumption of the network nodes to further enhance the network lifetime. DLA-BuS and DLA-MRF have different goals. Employing of them would highly depend on the application requirements. For example, in a highly risk environment such as a war zone, RAM would be better suited since simultaneous failure of adjacent nodes may normally take place. Generally, simultaneous node failures are very challenging since a part of the deployment area is subject to a major hazardous event (e.g., hit by a bomb). In the future, we plan to extend our approach to recover the partitioned network from k adjacent simultaneous nodes failures. Another plan that we will focus on is to evaluate the performance of the proposed approaches in a network of robots experimentally.

References

[1] C. Zhu, C. Zheng, L. Shu, G. Hang, A survey on coverage and connectivity issues in wireless sensor networks, *Journal of Network and Computer Applications* 35 (2012) 619-632.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, A Survey on Sensor Networks, *Communications Magazine* 40 (2002) 102-114.

[3] R. Kay, F. Mattern, The design space of wireless sensor networks, *Wireless Communications* 11 (2004) 54-60.

[4] Y. Lai, H. Chen, Energy-Efficient Fault-Tolerant Mechanism for Clustered Wireless Sensor Networks, *Computer Communications and Networks. 16th International Conference on IEEE* (2007) 272-277.

[5] F. Koushanfar, M. Potkonjak, A. Sangiovanni, Fault tolerance techniques for wireless ad hoc sensor networks, *Sensors, IEEE* 2 (2002) 1491-1496.

[6] P. Jiang, A New Method for Node Fault Detection in Wireless Sensor Networks, *Sensors, IEEE* 9 (2009) 1282-1294.

[7] J. Chen, M. Daz, L. Llopis, B. Rubio, J. M. Troya, 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* 34 (2011) 1225-1239.

[8] V. Ranga, M. Dave, A. K. Verma, A hybrid timer based single node failure recovery approach for WSANs, *Wireless personal communications* 77 (2014) 2155-2182.

[9] M. Younis, K. Akkaya, Strategies and techniques for node placement in wireless sensor networks: A survey, *Ad-Hoc Networks* 6 (20084) 621-655.

[10] K. Ozaki, K, Watanabe, K, Itaya, S, Hayashibara, N, Enokido, T, and Takizawa, M, A fault-tolerant model for wireless sensor-actor system. *Proceedings of the 20th international conference on advanced information networking and applications* Vienna, Austria; (2006).

[11] I. Akyildiz, I. Kasimoglu, Wireless sensor and actuator networks: Research challenges, *Ad-hoc Networks* 2 (2004) 351-367.

[12] S. Das, H, Liu, A. Kamath, A. Nayak, I. Stojmenovi, Localized movement control for fault tolerance of mobile robot networks,

Wireless Sensor and Actor Networks 248 (2007) 1-12.

[13] G. H. Lin, G. Xue, Steiner tree problem with minimum number of Steiner points and bounded edge-length, *Information Processing Letters* 69 (1999) 53-57.

[14] P. Basu, J. Redi, Movement control algorithms for realization of fault-tolerant adhoc robot networks, *Network* 18 (2004) 36-44.

[15] A. A. Abbasi, M. Younis, K. Akkaya, Movement-Assisted Connectivity Restoration, *Wireless Sensor and Actor Networks, IEEE transactions on parallel and distributed systems* 20 (2009) 1366-1379.

[16] M. Imran, M. Younis, AM. Said, H. Hasbullah, Volunteer-instigated connectivity restoration algorithm for wireless sensor and actor networks,*Proceedings of the IEEE International Conference on Wireless Communications, Networking and Information Security* 5 (2010) 679-683.

[17] K. Akkaya, F. Senel, A. Thimmapuram, S. Uludag, Distributed Recovery from Network Partitioning, *Movable Sensor/Actor Networks via Controlled Mobility, IEEE Transactions on Computers* 59 (2010) 258-271.

[18] A. A. Abbasi, K. Akkaya, M. A. Younis, Distributed connectivity restoration algorithm in wireless sensor and actor networks, InLocal Computer Networks, *32nd IEEE Conference* (2007) 496-503.

[19] M. Younis, S. Lee, S. Gupta, K. A. Fisher, Localized self-healing algorithm for networks of moveable sensor nodes, *Global Telecommunications Conference, 2008. IEEE GLOBECOM* (2008) 1-5.

[20] N. Tamboli, M. Younis, Coverage-aware connectivity restoration in mobile sensor networks, *Journal of Network and Computer Applications* 33 (2010) 363-374.

[21] A. Boudries, M. Aliouat, P. Siarry, Detection and replacement of a failing node in the wireless sensors networks, *Computers and Electrical Engineering* 40 (2013) 421-432.

[22] M. Imran, M. Younis, A. M. Said, H. Hasbullah, Localized motion-based connectivity restoration algorithms for wireless sensor and actor networks, *Journal of Network and Computer Applications* 35 (2012) 844-856.

[23] M. Jahanshahi, M. Maddah, N. Najafizadegan, Energy aware distributed partitioning detection and connectivity restoration algorithm in wireless sensor networks, *International Journal of Mathematical Modelling and Computations* 3 (2013) 71-82.

[24] N. Maalel, M. Kellil, P. Roux, A. Bouabdallah, Fast Restoration of Connectivity for Wireless Sensor Networks, *Internet of Things, Smat Space And Next Generation Networking Lecture Computer Science*, 7469 (2012) 401-412.

[25] K. S. Narendra, M. A. Thathachar, Learning automata: an introduction, *Courier Dover Publications* (2012).

[26] K. Wu, Y. Gao, F. Li, Y. Xiao, Light weight deploymen taware scheduling for wireless sensor networks, *Mobile Networks and Applications* 10 (2005) 837-852.

[27] K. S. Narendra, M. A. L. Thathachar, Learning automata a survey, *IEEE Transactions on Systems*, Man and Cybernetics 4 (1974) 323-334.

[28] M. E. Harmon, S. S. Harmon, Reinforcement Learning: A Tutorial, WL/AAFC, WPAFB Ohio, 45433, (1997).

[29] R. S. Sutton, A. G. Barto, Reinforcement learning: Introduction, *MIT Press* (1998).

[30] M. A. L. Thathachar, P. S. Sastry, A hierarchical system of learning automata that can learn the globally optimal path, *Information Science* 42 (1997) 743-766.

[31] B. R. Harita, Learning automata with changing number of actions, *IEEE Transactions on Systems, Man, and Cybernetics SMG* 17 (1987) 1095-1100.

[32] M. A. L. Thathachar, V. V. Phansalkar, Convergence of teams and hierarchies of

learning automata in connectionist systems, *IEEE Transactions on Systems, Man and Cybernetics* 25 (1995) 1459-1469 (1995).

[33] S. Lakshmivarahan, M. A. L. Thathachar, Bounds on the convergence probabilities of learning automata, *IEEE Transactions on Systems, Man, and Cybernetics* 6 (1976) 756-763.

[34] M. Esnaashari, M. R. Meybodi, M. Sabaei, A novel method for QoS support, *sensor networks, In CSICC* (2007) 740-747.

[35] N. Farajzadeh, M. R. Meybodi, Learning automata-based clustering algorithm for sensor networks, *CSICC*, (2007) 780-787.

[36] K. S. Narendra, M. A. L. Thathachar, On the behavior of a learning automaton in a changing environment with application to telephone traffic routing, *IEEE Transactions on Systems, Man, and Cybernetics* l0 (1980) 262-269.

[37] M. Jorgic, M. Hauspie, D. SimplotRyl, I. Stojmenovic, Localized algorithms for detection of critical nodes and links for connectivity in ad hoc networks, *In the 3rd IFIP Mediterranean Ad Hoc Networking Workshop* (2004) 1-12.

[38] R. Jaichandran,A. Anthony Irudhayara, J. Emerson Raj, Effective strategies and optimal solutions for Hot Spot Problem in *wireless sensor networks (WSN), In Information Sciences Signal Processing and their Applications* (2010) 389-392.

[39] J. Akbari Torkestani, M. Meybodi, An intelligent backbone formation algorithm for wireless ad hoc networks based on distributed learning automata, *In Computer Networks* 54 (2009) 826-843.

[40] J. Akbari Torkestani, Backbone formation in wireless sensor networks, *In Sensors and Actuators A: Physical* 185 (2012) 117-126.

[41] H. Beigy, M. R. Meybodi, A New Distributed Learning Automata Based Algorithm For Solving Stochastic Shortest Path Problem, In *JCIS*, (2002) 339-343.

[42] P. Crescenzi, V. Kann, A compendium of NP optimization problems, (1995).

Mohsen Jahanshahi, Senior Member, IEEE, is an Associate Professor at the department of computer engineering, Central Tehran Branch, Islamic Azad University, Tehran, Iran. Dr. Jahanshahi is also a member of Young Researchers and Elite club since 2012. His research interests include wireless sensor networks, cognitive networks, learning systems, mathematical optimization, and soft computing.

Mina Maddah was born in Qazvin, Iran, in 1987. She received the BSc. and MSc. degrees from Qazvin Islamic Azad University, Qazvin, Iran, in 2010 and 2015 respectively. She is currently working as computer network senior in Mobile Telecommunication Company of Iran (MCI). Her research interests include learning automata, interconnection networks, wireless Sensor networks and computer networks.

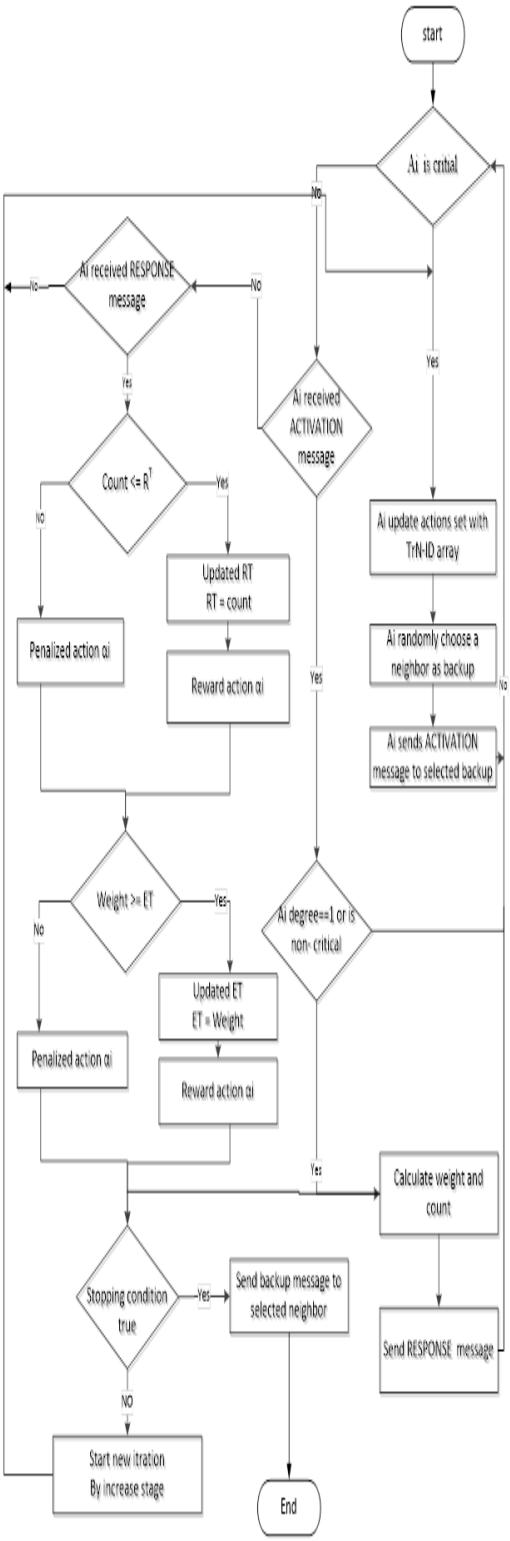

Figure 6: DLA-BuS flowchart

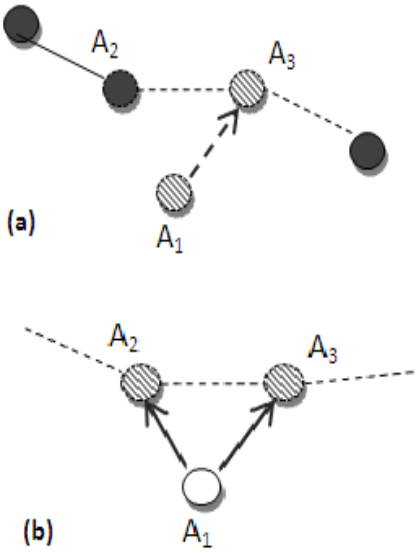

Figure 7: Example of the simultaneous failure of two adjacent nodes (a) the failure of two adjacent critical nodes with the same backup, (b) failure of a critical node and its backup.

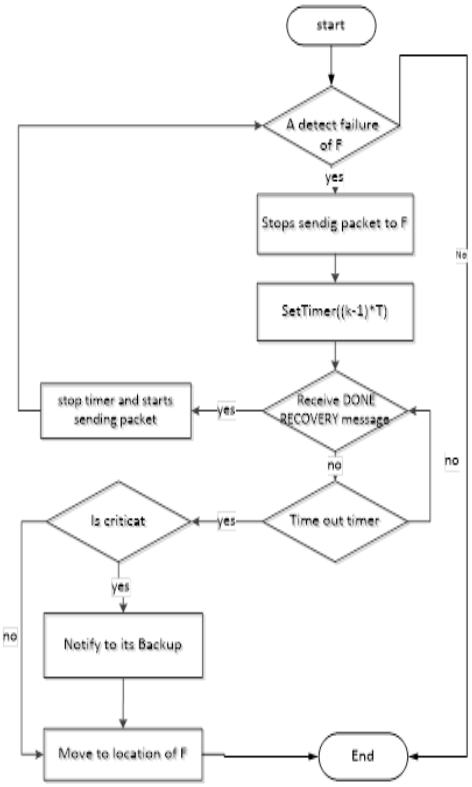

Figure 8: DLA-MRF flowchart

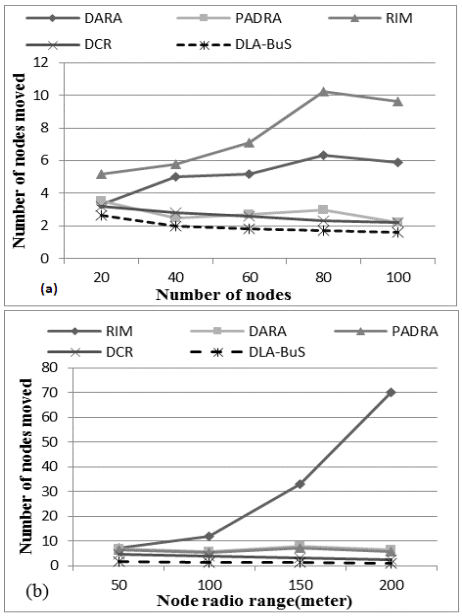

Figure 9: The number of nodes moved during the recovery, while varying (a) the network size (b) and radio range.

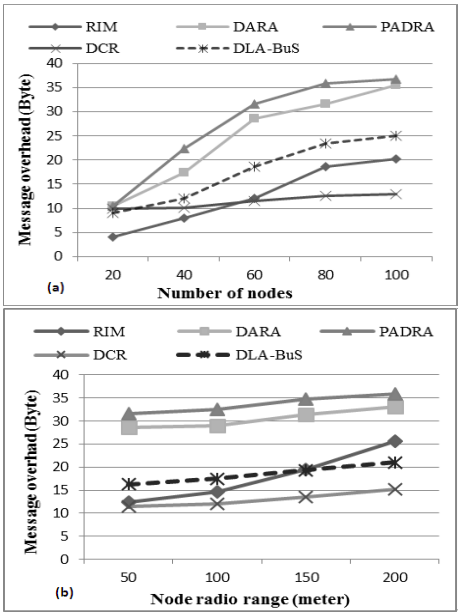

Figure 11: Message overhead during the recovery, while varying (a) the network size (b) and radio range.

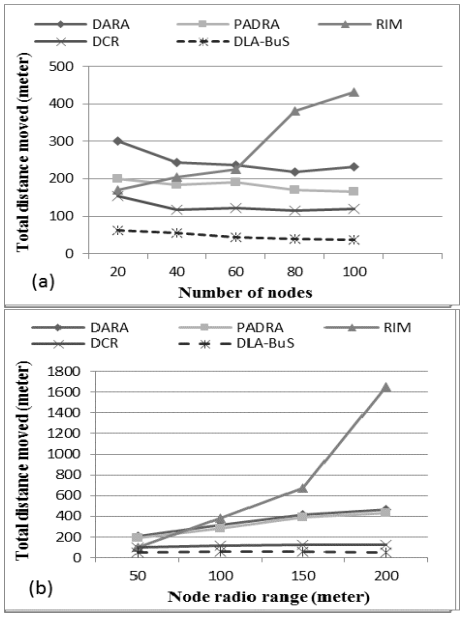

Figure 10: Total distance moved during the recovery, while varying (a) the network size (b) and radio range.

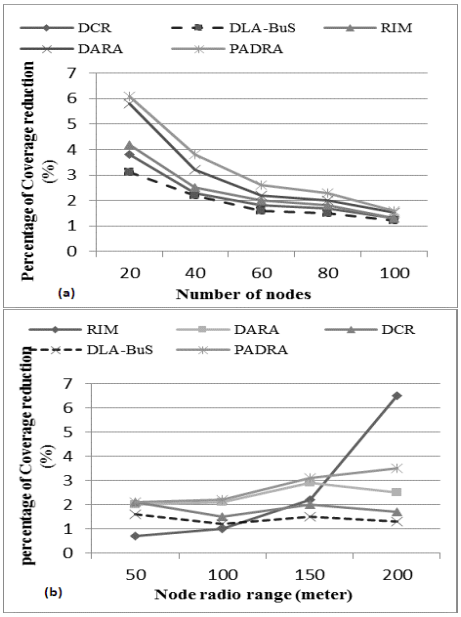

Figure 12: Percentage of coverage reduction during the recovery process, while varying (a) the network size and (b) radio range.

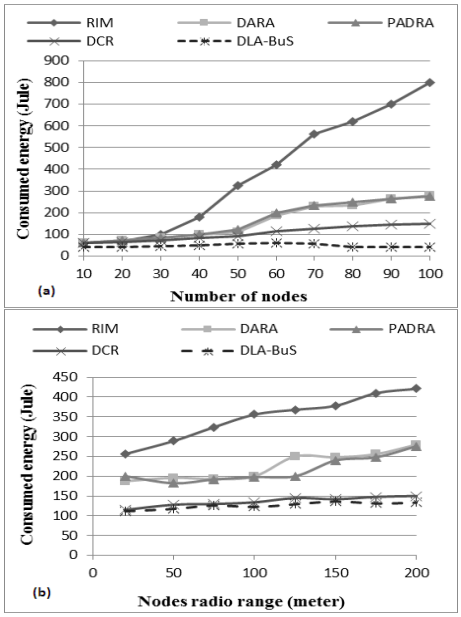

Figure 13: Percentage of consumed energy during the recovery, while varying (a) the network size and (a) radio range.

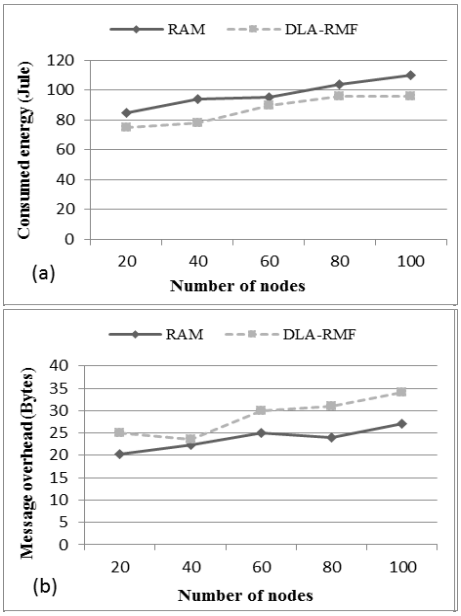

Figure 15: Consumed energy and (b) message overhead during the recovery of DLA-MRF, while varying the network size.

Parameter	value
L_{HB} (Heartbeat packet length)	5 Byte
S (movement speed of each node)	1 m/s
$= r_c$ *(Communication range)* d_{max}	50 m
R (Heartbeat's packet rate)	250 Bps

Figure 14: Simulation parameters value

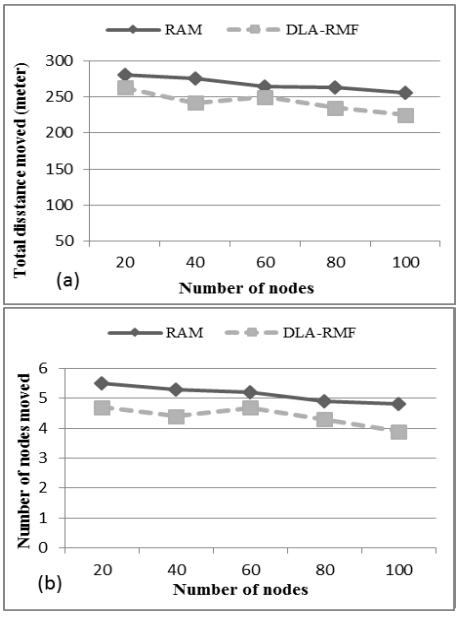

Figure 16: (a) Total distance and (b) number of nodes moved during the recovery of DLA-MRF, while varying the network size.

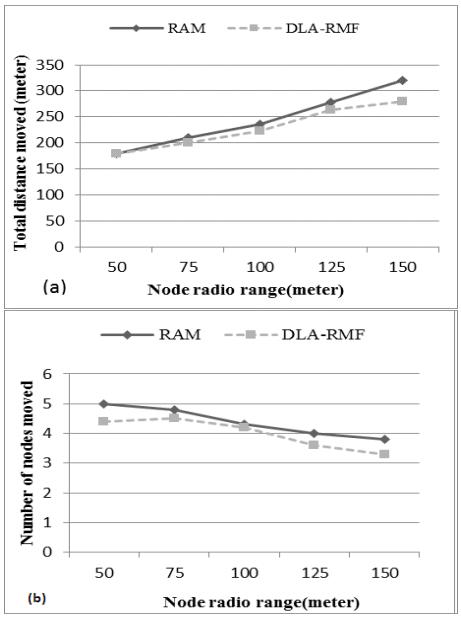

Figure 17: Consumed energy and (b) message overhead during the recovery of DLA-MRF, while varying the network size.