Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2012, Article ID 273792, 9 pages
doi:10.1155/2012/273792

Research Article

Application-Oriented Fault Detection and Recovery Algorithm for Wireless Sensor and Actor Networks

Jinglin Du,[1,2] Li Xie,[1] Xiaoyan Sun,[2] and Ruoqin Zheng[2]

[1] *Department of Computer Science and Technology, Nanjing University, Nanjing 210093, China*
[2] *Department of Electronics and Information Engineering, Nanjing University of Information Science and Technology, Nanjing 210044, China*

Correspondence should be addressed to Xiaoyan Sun, sun.xiaoyan.1989@163.com

Received 30 July 2012; Accepted 17 September 2012

Academic Editor: Wenzhong Li

Recent years have witnessed a growing interest in applications of wireless sensor and actor networks (WSANs). In WSANs, maintaining interactor connectivity is of vital concern in order to reach application level. Failure of a critical actor may partition the inter-actor network into disjoint segments. This paper proposed an application-oriented fault detection and recovery algorithm (AFDR), a novel distributed algorithm to reestablish connectivity. AFDR identifies critical actors and designates backups for them. A backup actor detects the critical node failure and initiates a recovery process via moving to the optimal position. The purpose of AFDR is to satisfy application requirements, reduce recovery overhead, and limit the impact of critical node failure on coverage and connectivity to the utmost. The effectiveness of AFDR is validated through simulation experiments.

1. Introduction

One of the most attractive and important parts of The Internet of Things is known as the wireless sensor and actor networks (WSANs) [1]. WSANs are finding applications in many domains such as disaster management, homeland security, battlefield reconnaissance, space exploration, search and rescue, and so forth. A WSAN consists of numerous miniaturized sensor nodes and fewer actor nodes [2]. The sensor nodes probe their surroundings, measure ambient conditions, and transmit the collected data to one or multiple actors for processing, making decisions, and responding to events of interest.

In WSANs, connectivity of the network is crucial throughout the lifetime of the network in order to meet the desired application-level requirements. As far as WSANs are concerned, in most application setups, actors need to coordinate with each other in order to share and process the sensors' data and plan an optimal response. In such connected WSANs, failure of one or multiple nodes may cause the loss of other nodes or communication links, partitioning of the network if alternate paths among the affected nodes are not available, and stopping the actuation capabilities of the node. Such a scenario will not only hinder the nodes' collaboration but also has very negative consequences on the considered applications. Therefore, WSANs should be able to tolerate the failure of mobile nodes and self-recover from them in a distributed, timely, and energy efficient manner: first, the recovery should be distributed since these networks usually operate autonomously and unattended. Second, rapid recovery is desirable in order to maintain the responsiveness to detected events. And finally, the energy overhead of the recovery process should be minimized to extend the lifetime of the network [3].

In this paper, we present an application-oriented fault detection and recovery algorithm (AFDR) to determine possible partitioning in advance and self-restore the connectivity in case of such failures with minimized node movement and message overhead. Since partitioning is caused by the failure of a node which is serving as a cut vertex (critical node) [4], each node determines whether it is a cut vertex or not in a distributed manner (Arrival Message Matrix (AMM)). Each node in the network periodically sends out heartbeat messages and decides whether it is a cut vertex based on the received feedback. Once such cut vertex nodes are determined, each node designates the appropriate neighbor called backup to handle its failure. The backup detects any failure of the cut vertex and chooses a suitable node (a leaf node or itself) to replace the failed node. The alternative node need not move to the exact position directly where the failed actor is; it just moves to an optimal position to administer more sensors. The goal of AFDR is to ensure the coverage of actors and reduce the mobility and communication overhead caused in the connectivity recovery process.

2. System Model and Problem Statement

AFDR is applicable to WSANs that consist of sensors and actors. Sensors detect and report events of interest to one or multiple actors. Actors receive reports from sensors and process and collaborate with each other to plan an optimal coordinated response. The communication range of an actor refers to the maximum Euclidean distance that its radio can reach. An actor has two radios for sensor-actor and actor-actor communications. To simplify analysis, nodes are assumed to have the same communication range. Both sensors and actors are deployed randomly in an area of interest. After deployment, actors are assumed to discover each other and form a connected inter-actor network [3]. An actor is assumed to be able to move on demand and before moving it informs its backup so that it may not be wrongly perceived as faulty.

In order to provide qualified services, many applications require overlays [5] to guarantee reliability and avoid network failures [6]. For a large-scale distributed system, it is often not cost-effective to protect all the nodes [7]. While node failures may affect network coverage [8] and connectivity, this paper focuses on both, especially on maintaining the latter when a node is lost. The impact of an actor's failure depends on the position of that actor in the network topology. For example, losing a noncritical node does not affect inter-actor connectivity. Meanwhile, the failure of critical node will partition the network into disjoint segments. AFDR pursues actor relocation to recover from critical node failures. We consider one failure at a time and assume that no node fails during the recovery of another.

3. Related Work

In most WSAN applications, actors establish a connected inter-actor topology in order to coordinate with each other on an optimal response and synchronize their operations. However, the harsh environment which WSAN operates in makes actors vulnerable to physical damage and component malfunction. An actor failure may partition the inter-actor network into disjoint segments and consequently hinders inter-actor interaction. Numerous schemes have been pursued recently for repairing network connectivity in partitioned WSANs.

The main idea is to identify and relocate some of the nodes. Employing node mobility to repair damaged network topologies has started to attract attention. Most of the existing approaches in the literature are purely reactive [9], which can be categorized into block [10] and cascaded movements. Block movement often requires a high prefailure connectivity in order for the nodes to coordinate their response. It often becomes infeasible in the absence of higher level of connectivity. Therefore, cascaded node movement that can be further categorized based on network state information that nodes are assumed to maintain is exploited. Some approaches like DARA [2] and PADRA [3] require each actor to maintain two-hop neighbors. One of the neighbors of the failed node is picked to initiate the recovery process such that the movement overhead and the number of messages are minimized. While DARA designates the node with the least node degree as the recovery initiator and strives to restore connectivity lost due to failure of cut vertex, PADRA identifies a connected dominating set to determine a dominatee node in order to detect cut vertices. The dominatee does not directly move to the location of the failed node. Nonetheless, they use distributed algorithm and their solution still requires 2-hop neighbor's information that increases messaging overhead.

Some localized algorithms require only 1-hop neighbor's positional information at the expense of lower accuracy of cut vertices identification. Basically, some nodes are marked as critical while they are not cut vertices. However, no critical node will be missed. DCR [11] and RAM [12] employ a simple localized cut vertex detection procedure that runs on each node in a distributed manner to determine locally whether a node is critical or not.

Akkaya et al. introduced a mutual exclusion mechanism called MPADRA [13] in a localized manner. Both RAM and MPADRA can handle simultaneous failure of multiple actors but MPADRA differs from RAM in multiple aspects. First, MPADRA requires a mutual exclusion mechanism to avoid race conditions. Second, MPADRA reserves the nodes on the path in advance before actual relocation. On the other hand, RAM designates distinct backups and does not engage relocating nodes beforehand. Third, MPADRA maintains 2-hop network state information and requires primary and secondary failure handlers for each dominator.

Younis et al. proposed an algorithm called RIM [14]. When a node fails, its neighbors move inward toward its position so they can connect with each other. The rationale is that these neighbors are the ones directly impacted by the failure, and when they can reach each other again, the network connectivity is restored to its prefailure status. The relocation procedure is recursively applied to handle any nodes that get disconnected when one of their neighbors moves.

Another algorithm VCR [15] exploits the fact that some neighbors of the failed node are not using their full communication range and would thus be able to reach more distant nodes than the failed actor. In VCR, each actor maintains a list of 1-hop neighbors and monitors their heartbeats. The failure of an actor is detected through missing heartbeats. The recovery process consists of two phases. At first, volunteer actors are identified. Then the topology repair is performed through uncoordinated relocation of the volunteer actors while exploiting partially utilized transmission range and actor diffusion.

4. Connectivity Restoration Algorithm

4.1. Cut Vertex Identification. Every node in the system considers itself as a cut vertex candidate and initializes a cut vertex detection process. The node that has zero or only one connection does not need to initialize this process since it can not be a cut vertex. At the beginning, the candidate assigns a unique numerical identifier for its connections which is called the connection number. For example, if a candidate has c connections, it will label them from 1 to c.

At first, the candidate sends a component probe message to each of its neighbors. The message contains the following elements: the candidate's ID, a timestamp, a TTL threshold, and the number of connections that connects this neighbor with the candidate. So each node in the system has a connection list. The format of the entry for a candidate in the connection list is "candidate ID, timestamp, TTL threshold, connection number 1, connection number 2, . . .". Upon receiving a message, each node deals with it when one of the following situations happens.

(a) If the node has already received the message, or the message is old, the node simply just drops the message.

(b) If there is no entry for the candidate that issues this message, the node creates an entry for it.

(c) If the timestamp in the received message is newer than the one stored in the connection list, the node replaces the old timestamp and connection numbers stored in the connection list with the new ones.

(d) If the timestamp in the received message is the same as the one stored in the connection list but the connection number of the message differs from the old one, the node adds the new connection number to the entry and sends an arrival message back to the candidate.

So each candidate maintains an arrival list whose format is similar to the connection list: "node ID, timestamp, connection number 1, connection number 2, . . .". That is to say, a node does not send any arrival messages until its probe messages contain at least two different connection numbers [3].

According to the arrival list, each candidate maintains a c-by-c binary matrix, where c is the number of connections the candidate has. The row or column numbers of the binary matrix represent the connection number and the matrix is called the candidate's AMM (Arrival Message Matrix). Even though the node with high degree has a big matrix, the storage overhead is so small when compared to the movement overhead in the recovery process that it can be ignored. For an entry (x, y) in the matrix, where x is the row number and y is the column number, if the corresponding connection number of x and y can be found in the same entry of the arrival list, the value of this matrix entry is set to 1. Otherwise, the value is set to 0. In other words, if any node has sent back an arrival message containing connection numbers x and y to the candidate, 1 is set in the (x, y) and (y, x) entries of the candidate's AMM. After waiting a defined period of time, the candidate forms an undirected graph coming from the matrix representation. The vertices of this graph are corresponding to the candidate's connections. We call this graph the candidate's AMM graph. Node x and node y are connected in the AMM graph if and only if the value of the AMM entry (x, y) is 1. If the candidate's AMM graph has more than 1 component (a connected graph), then we do a local search among the two-hop neighbors of the candidate. When this candidate is removed from the system, the rest of the actors still stay connected and the candidate is not a cut vertex. Instead, we determine that the candidate is a cut vertex.

4.2. Backup Selection and Failure Detection.

4.2. Backup Selection and Failure Detection. Once the cut vertices are identified, the next step is to select and designate appropriate neighbors as backups for them. The purpose of the prenomination of backup nodes is to instantaneously react to the failure of cut vertex and avoid the possible network partitioning caused by such a failure.

4.2.1. Selection of Backup. The actors maintain minimum state information (i.e., 1-hop neighbors) to avoid extra overhead of messaging. Since neighbors become disconnected when a critical actor fails, backup actors are determined and notified before a failure of critical nodes takes place. A node can serve as backup for multiple actors. The selection of a backup among 1-hop neighbors is based on the following ordered criteria.

Neighbor Position (NP). As discussed above, each actor determines whether it is critical or noncritical depending on the position of that node in the topology. A noncritical neighbor actor is more suitable for backup because it will limit the scope of recovery that ultimately reduces the impact on application, coverage, connectivity, and incurred overhead.

Actor Flow (AF). If a noncritical node is available in the 1-hop neighborhood of the primary actor, we choose the actor with the smallest flow package from sensors charged by it and other actors; in other words, the one which receives the least data packets from other nodes, as the cut vertex's backup.

Flow package is divided into two categories.

(i) Actor-actor flow package (AAFP): this flow package is transferred between an actor and another actor.

(ii) Actor-sensor flow package (ASFP): this flow package is transferred between an actor and a sensor. If the number of these packages for an actor is large, it indicates that sensors under the jurisdiction of the actor is more, so the actor can play a greater role for the application system.

Actor Degree (AD). The impact of moving a node that has many neighbors will be significant. Thus, if a noncritical node is not available in the neighborhood, AFDR favors replacing the failed actor with the neighbor that is a strongly connected critical node (with high degree) because there is more probability to have noncritical nodes in the neighborhood. This will limit the scope of cascaded relocation and thus lower the recovery overhead.

Take the circumstance in Figure 1 as an example. Green pentagons represent the actors while yellow circles stand for sensors. Among the 1-hop neighbors of actor 6, actor 7 is not a critical node, so it is best suited to be the backup. If we remove actor 7, in the rest of the neighbors of actor 6, actor 8 and actor 9 will have the same AD, which will be smaller than actor 3. At this point, we continue to compare their AF. The one whose AF is smaller will be chosen as a backup of actor 6. Figure 2 shows the pseudocode of backup selection of AFDR algorithm.

4.2.2. Failure Detection. Neighbor actors exchange heartbeat messages as part of their network operation to update their status. The chosen backup actors are notified via these messages. Once an actor receives backup notification, it starts monitoring the cut vertex through heartbeats. Missing a number of consecutive heartbeats is perceived by backup as a failure of the cut vertex.

4.3. Failure Recovery. Despite the fact that failure of a node which is not a cut vertex will not cause any problems to the inter-actor connectivity, it can create other problems such as forming coverage holes, disrupting the data collection from that particular region, and so forth. In such cases, depending on the application-level requirements, these problems need to be handled. We would like to note, however, that handling such problems is out of scope of this paper. We only focus on restoration of inter-actor connectivity when a cut vertex node fails.

4.3.1. Definition: Optimal Position (OP). AFDR defines an optimal position, when the actor failure, its replacement node does not need to move to the original location where the failure node is, the new location of this alternative node has the following characteristics: it can communicate with the surrounding sensors as many as possible, that is, the more ASFP it received, the better; the distance it moved from its original location to the optimal position should be as short as possible to reduce overhead. If this location meets the above conditions, we call it the optimal position.

4.3.2. OP Computation Model. We proposed a model to calculate the OP. For the convenience of description, we take the following circumstance as an example. In Figure 3, actor 3 serves as a backup for the node F. When F fails, it partitions the network into two parts. Actor 1 and actor 2 are respectively, the nearest nodes to F in the two subnets. Communication range of actor 1 is shown as circle A and simultaneously, circle B represents the communication range of actor 2. These two circles' intersections are M and N. C is the overlap area of actor 1 and actor 2's communication range. According to the location of the backup, we determine the optimal position.

(i) If actor 3 is located in A but not in C, do a connection between actor 3 and actor 2; the intersection of the line and circle B is the OP. If actor 3 is located in B but not in C, just use the same method to pursue the OP.

(ii) If actor 3 is located in C, the location of actor 3 itself is OP.

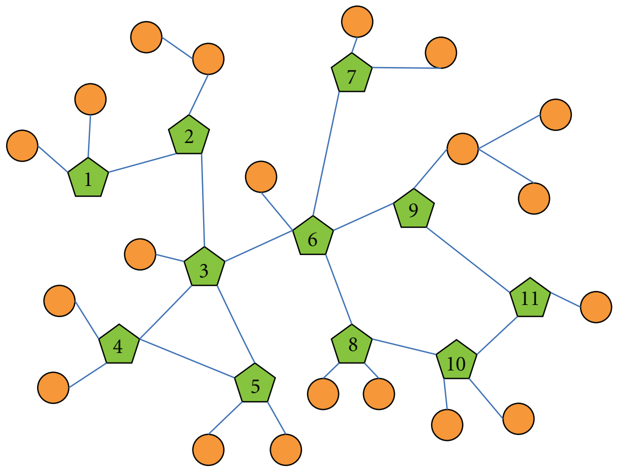
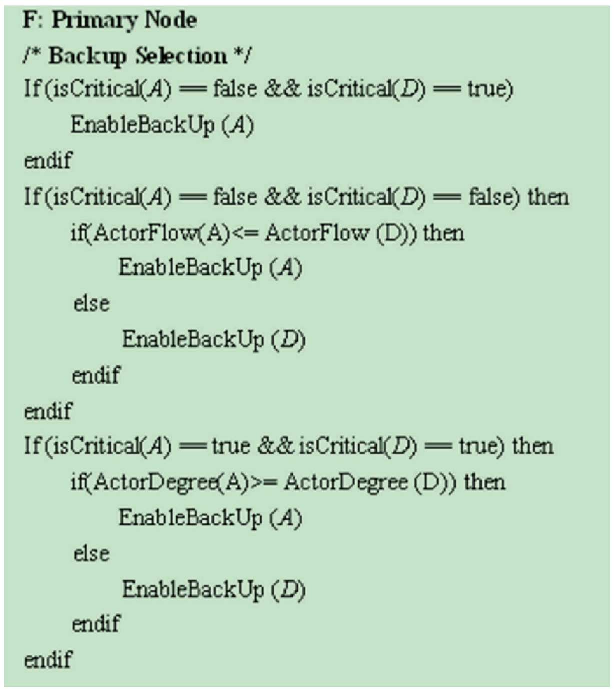


Figure 1: A connected interactor network with critical and noncritical actors.



```
F: Primary Node
/* Backup Selection */
If(isCritical(A) == false && isCritical(D) == true)
    EnableBackUp (A)
endif
If(isCritical(A) == false && isCritical(D) == false) then
    if(ActorFlow(A)<= ActorFlow (D)) then
        EnableBackUp (A)
    else
        EnableBackUp (D)
    endif
endif
If(isCritical(A) == true && isCritical(D) == true) then
    if(ActorDegree(A)>= ActorDegree (D)) then
        EnableBackUp (A)
    else
        EnableBackUp (D)
    endif
endif
```

Figure 2: Pseudocode of backup selection of AFDR algorithm.

(iii) If actor 3 is located neither in circle A nor circle B, either M or N will be the OP. Choose the one that is closer to actor 3.

The predesignated backup actor immediately initiates a recovery process once it detects the failure of the cut vertex. There are four scenarios that may happen. First, if the backup is noncritical then it simply replaces the cut vertex and moves to the OP so that recovery is complete. Second, if the backup actor is critical and has a leaf node in its neighborhood, the leaf node just replaces the cut vertex and moves to the OP. Third, if the backup actor is critical then it checks whether the failed node was also its backup or not; when

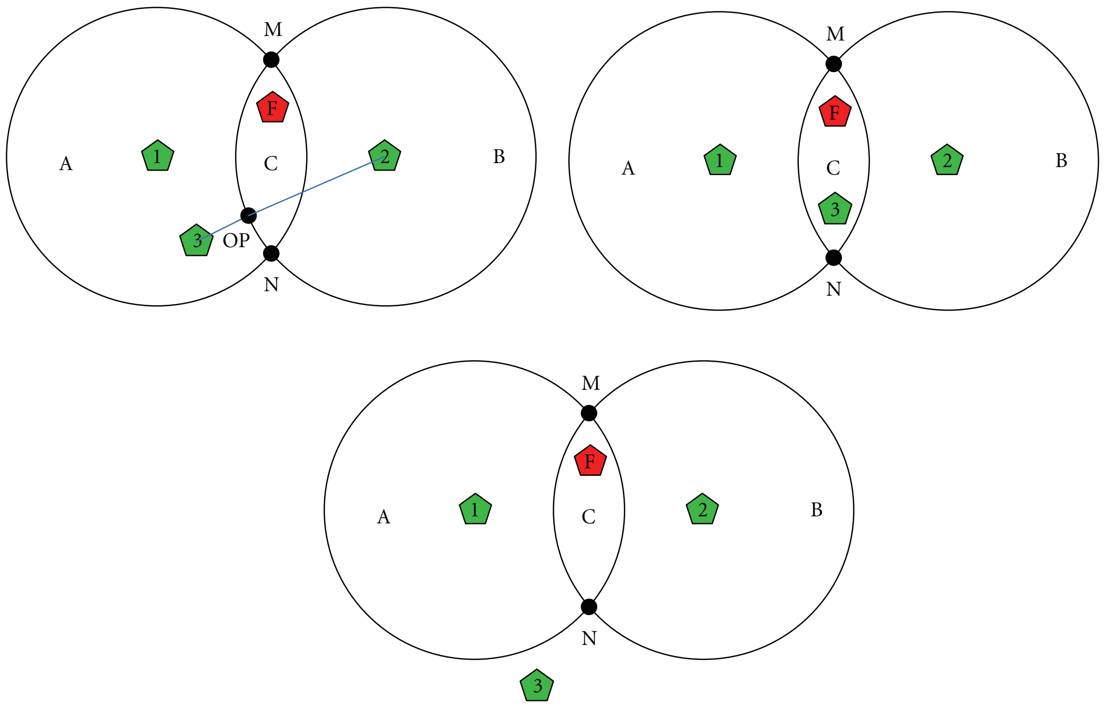


FIGURE 3: OP calculation.

the two nodes are backup for each other, the backup actor appoints another backup using the same criteria as specified in the preceding section and moves to the OP. Finally, if the predesignated backup actor is critical and its backup is alive then it just sends a movement notification message to the backup and moves to the OP. Since, in scenario 3 and scenario 4, moving a critical backup actor further partitions the network, a cascaded relocation may be triggered.

In Figure 4, the two circles, respectively, represent the communication range of actor 3 and actor 9. Obviously, actor 7 is the backup of actor 6. When actor 6 fails, based on the above OP calculation method, we can get the optimal position for actor 7. As long as actor 7 is a leaf node, it directly moves to the OP to ensure the connectivity of this network. Figure 5 indicates the location of actor 7 after the recovery progress is finished.

Figure 6 shows the pseudocode of the recovery process of AFDR algorithm.

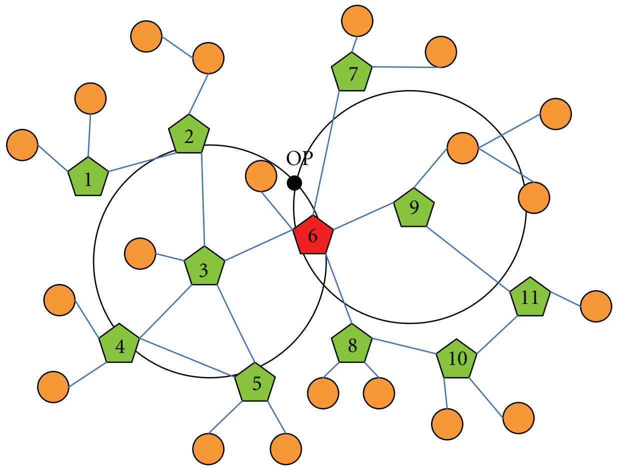


FIGURE 4: Original topology when an actor fails.

5. Algorithm Analysis

5.1. Simulation Setup and Performance Metrics. In the simulation experiments, we have created inter-actor topologies consisting of a varying number of nodes (20–100). Nodes are randomly placed in an area of 1000 m × 600 m with no obstacles that hinder a node from moving to a new position. We have varied the transmission range of actors from 50 to 125 m so that the topology becomes strongly connected. The following parameters were used to vary the WSAN configuration in the experiments.

(i) The number of deployed nodes (N) in the network affects the node density and the inter-actor connectivity.

(ii) The node communication range (r) influences the network connectivity and highly affects the recovery overhead in terms of the traveled distance and the number of involved actors.

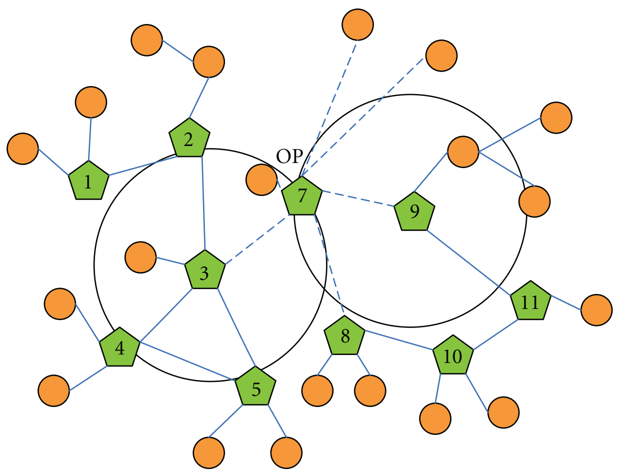


Figure 5: Recovery process when an actor fails.

We assessed the performance of AFDR by using the following metrics.

(i) The total traveled distance: it gives the total distance caused by all node movements involved in the recovery. This metric gauges how much energy will be consumed by the whole network due to the mechanical movements of nodes.

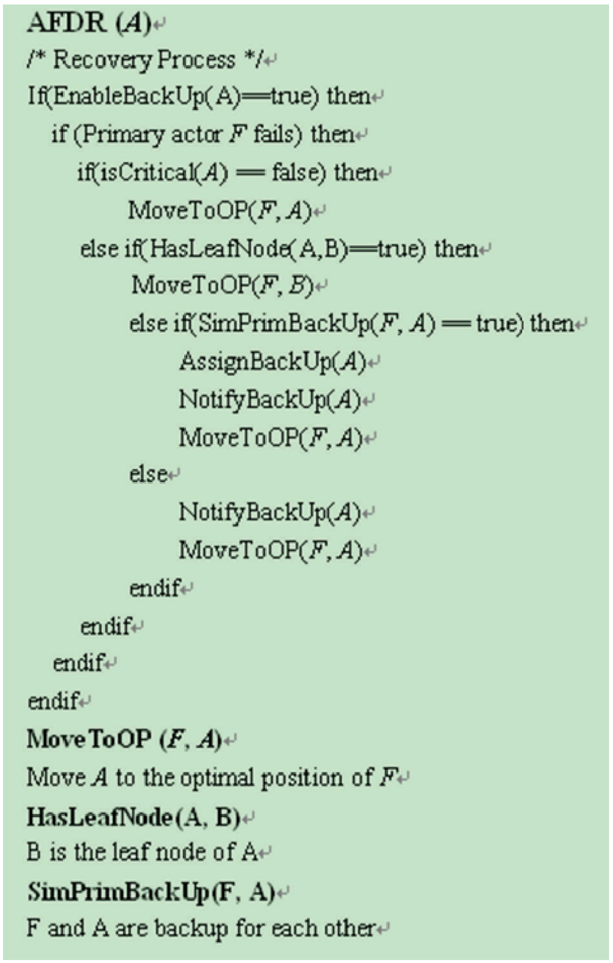
(ii) The number of messages exchanged among nodes: this metric also indicates the energy dissipation and recovery overhead.

(iii) The percentage of coverage reduction: although connectivity is the main objective of a recovery algorithm, node coverage is important for many setups. The loss of a node usually has a negative impact on coverage. This metric captures the loss of coverage resulting from the node movements.

(iv) Average node degree: it measures the level of inter-actor connectivity and availability of alternative paths after the recovery is complete.

The performance of AFDR is compared to DARA and RIM. Both DARA and RIM are reactive approaches and do not provision for recovery ahead of time. Like AFDR, DARA and RIM are distributed algorithms and exploit node relocation to recover from node failure. However, they have differences in the procedure. When a node fails, DARA selects a best candidate among its 1-hop neighbors and replaces it. This algorithm uses a recursive method to tolerate connectivity loss due to movement, that is, the chosen candidate will be replaced with one of its neighbors and so on. On the other hand, RIM moves all the 1-hop neighbors towards the failed node until they become connected again. As a result, when growing the communication range, the performance of RIM significantly worsens.

5.2. Results and Analysis. The number of nodes has been set to 20, 40, 60, 80, and 100. The communication range of actors is changed among 50, 75, 100, and 125. When changing the node count, "*r*" is fixed at 100 m; when changing the communication range, "*N*" is set to 100. The results of individual experiments are averaged over 50 trials.



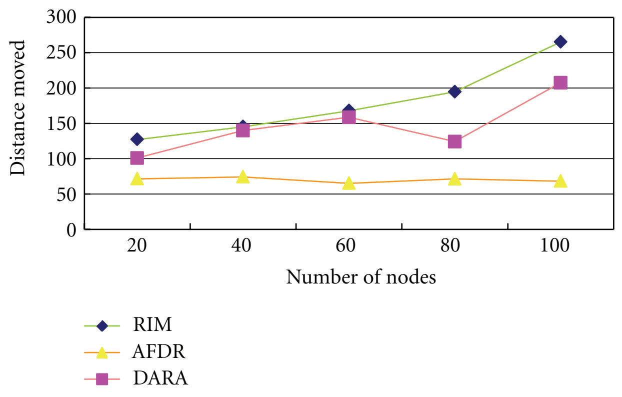
```
AFDR (A)
/* Recovery Process */
If(EnableBackUp(A)==true) then
  if (Primary actor F fails) then
    if(isCritical(A) == false) then
        MoveToOP(F, A)
    else if(HasLeafNode(A,B)==true) then
        MoveToOP(F, B)
        else if(SimPrimBackUp(F, A) == true) then
            AssignBackUp(A)
            NotifyBackUp(A)
            MoveToOP(F, A)
        else
            NotifyBackUp(A)
            MoveToOP(F, A)
        endif
    endif
  endif
endif
MoveToOP (F, A)
Move A to the optimal position of F
HasLeafNode(A, B)
B is the leaf node of A
SimPrimBackUp(F, A)
F and A are backup for each other
```

Figure 6: Pseudocode of the recovery process of AFDR algorithm.

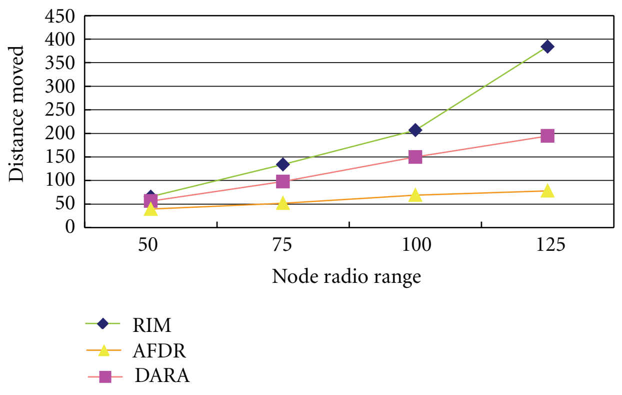
5.2.1. Total Traveled Distance. Figure 7 shows the distance traveled by all nodes until the connectivity is restored. AFDR obviously outperforms the existing approaches DARA and RIM.

As these two graphs in the figure indicate, the total node traveled distance of AFDR has not changed drastically even with higher node density and communication range. This is because AFDR strives to avoid moving critical nodes that causes further partitioning and requires cascaded relocations. It performs cascaded relocations only when noncritical nodes in the neighborhood of the failed actor are not available. RIM moves all the 1-hop neighbors towards the failed node until connectivity is reestablished. That is why its curve in the graph is steep.

5.2.2. Number of Messages Exchanged. Figure 8 presents the communication overhead when the network size and radio range take different values. As the figure indicates, AFDR begets less messaging overhead than DARA and RIM and

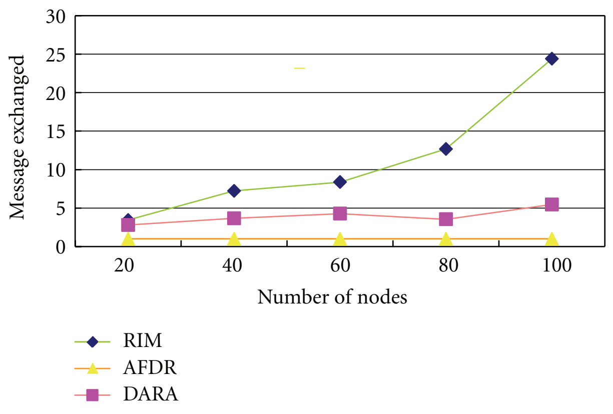


(a)

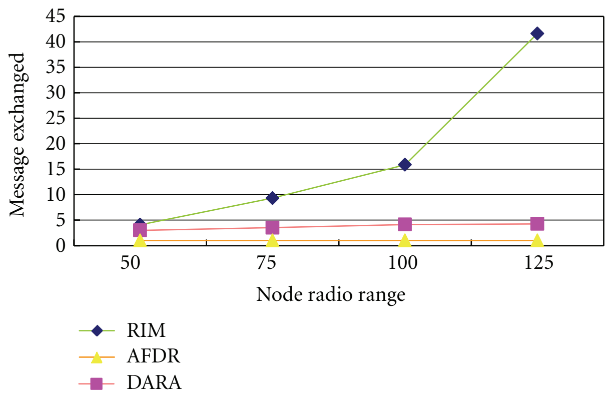


(b)

Figure 7



(a)



(b)

Figure 8

remains unchanged. Unlike DARA and RIM, AFDR strives to involve noncritical nodes in the recovery which limits the need of cascaded movement and thus reduces the number of notification messages. Therefore, AFDR makes best use of these messages, since most of the backup nodes are noncritical and they are not required to send any message. The average number of exchanged messages sent by AFDR in Figures 8(a) and 8(b) are less than 1. On the other hand, Figure 8 shows that the communication overhead in RIM grows rapidly for a higher actor density and long radio range.

5.2.3. Coverage Reduction Percentage. Figure 9 shows the impact of node failure on coverage, measured in terms of percentage of coverage reduction while changing N and r. Figure 9(a) indicates that the difference of curve trend among the three approaches is not too much as increasing the node number. Although increasing the node density helps, DARA and RIM still do not match AFDR's performance.

The advantage of AFDR in terms of coverage is obviously due to the limited scope of node relocation, which causes a coverage loss at the network periphery. In Figure 9(b), the advantage of AFDR highlights out as increasing the radio range. However, the performance of RIM worsens when growing the radio range above 100 m. With the increased value of r, the network becomes more connected and the number of neighbors of the failed node grows. RIM moves nodes inwards making the area around the failed node to be more crowded, thus leaves uncovered parts at the network periphery and causes a great loss of coverage.

5.2.4. Average Node Degree. Figure 10 shows the average actor degree and coverage in the network after connectivity is restored by all approaches. As showed in the figure, the data in the four graphs are basically the same. Figures 10(a)–10(d) confirm that AFDR does not have disadvantages in maintaining network connectivity.

Through the above analysis, we find that AFDR has advantages in reducing movement and communication overhead and keeping connectivity, which can make the life expectancy of the entire network longer. In other words, it will be easier to meet the application-level requirements.

6. Conclusion

In this paper, we have presented an application-oriented fault detection and recovery algorithm that focus on application-level concerns while recovering from critical node failure. The issued AFDR uses AMM to identify critical actors and

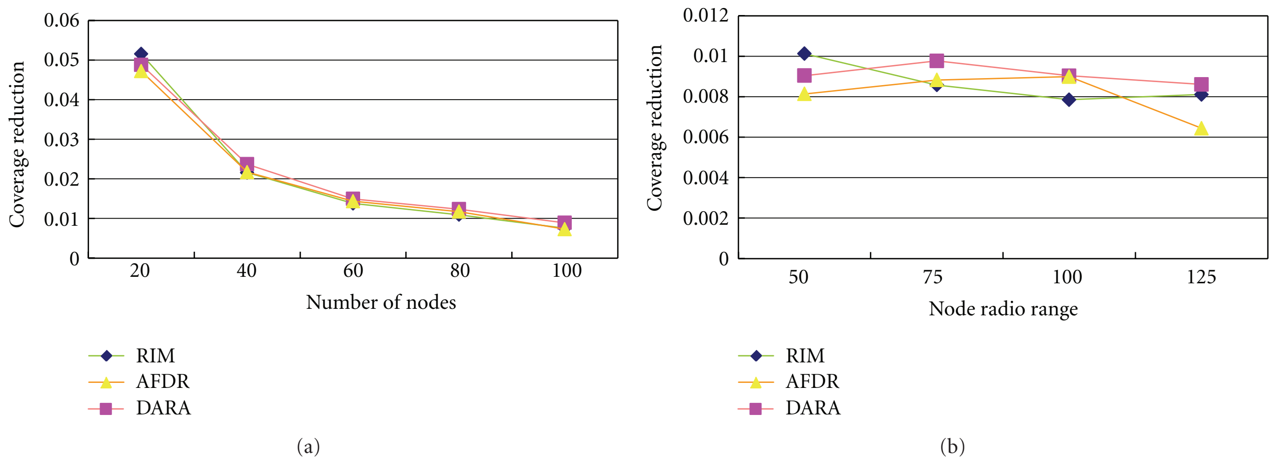


Figure 9

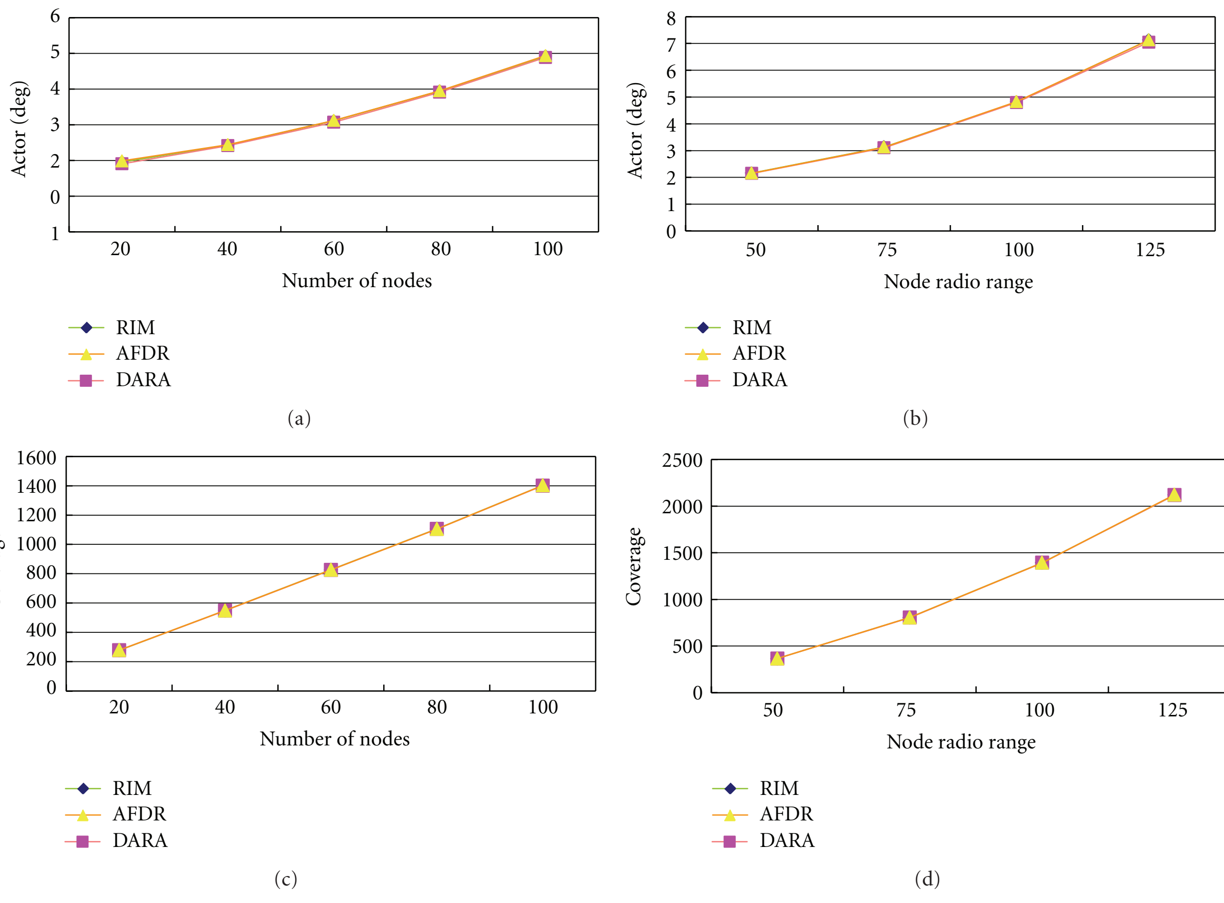


Figure 10

designates backup for them according to actor flow and actor degree. It tries to choose noncritical neighbors as backups to reduce the movement overhead and highly connected nodes as backups in order to limit the scope of cascaded relocation and nodes receiving smallest flow package as backups to minimize the communication overhead. In the recovery process, it finds an optimal position for backups to restore the inter-actor connectivity. The simulation results have proved the effectiveness of AFDR compared to contemporary recovery approaches in terms of satisfying application requirements, reducing recovery overhead and limiting the impact of critical node failure on coverage and connectivity.

In the future, our study can focus on how to identify cut vertex from a large amount of nodes in WSANs more quickly.

Acknowledgments

This work was supported by the 973 Program of China (nos. 2006CB303000 and 2009CB320-705), the National Natural Science Foundation of China (nos. 60873026 and 61021062), the National Science and Technology Support Program of China (no. 2012BAK26B02), the Science and Technology Support Program of Jiangsu Province (nos. BE2010178 and BE2011195), and the Industrialization of Science Program for University of Jiangsu Province (no. JH10-3).

References

[1] I. F. Akyildiz and I. H. Kasimoglu, "Wireless sensor and actor networks: research challenges," *Ad Hoc Networks*, vol. 2, no. 4, pp. 351–367, 2004.

[2] A. Abbasi, K. Akkaya, and M. Younis, "A distributed connectivity restoration algorithm in wireless sensor and actor networks," in *Proceedings of the 32nd IEEE Conference on Local Computer Networks (LCN '07)*, pp. 496–503, Dublin, Ireland, October 2007.

[3] K. Akkaya, A. Thimrnapuram, F. Senel, and S. Uludag, "Distributed recovery of actor failures in wireless sensor and actor networks," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC '08)*, pp. 2480–2485, Los Vegas, Nev, USA, April 2008.

[4] X. Liu, L. Xiao, A. Kreling, and Y. Liu, "Optimizing overlay topology by reducing cut vertices," in *Proceedings of the 16th Annual International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '06)*, May 2006.

[5] Y. Lin, B. Liang, and B. Li, "Data persistence in large-scale sensor networks with decentralized fountain codes," in *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM '07)*, pp. 1658–1666, May 2007.

[6] V. N. Padmanabhan, H. J. Wang, and P. A. Chou, "Resilient peer-to-peer streaming," in *Proceedings of the IEEE International Conference on Network Protocols (ICNP '03)*, 2003.

[7] Y. He, H. Ren, Y. Liu, and B. Yang, "On the reliability of large-scale distributed systems—a topological view," in *Proceedings of the 37th International Conference on Parallel Processing (ICPP '08)*, pp. 165–172, September 2008.

[8] G. Wang, G. Cao, T. La Porta, and W. Zhang, "Sensor relocation in mobile sensor networks," in *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '05)*, pp. 2302–2312, March 2005.

[9] N. Tamboli and M. Younis, "Coverage-aware connectivity restoration in mobile sensor networks," in *Proceedings of the IEEE International Conference on Communications (ICC '09)*, Dresden, Germany, June 2009.

[10] P. Basu and J. Redi, "Movement control algorithms for realization of fault-tolerant ad hoc robot networks," *IEEE Network*, vol. 18, no. 4, pp. 36–44, 2004.

[11] M. Imran, M. Younis, A. M. Said, and H. Hasbullah, "Partitioning detection and connectivity restoration algorithm for wireless sensor actor networks," in *Proceedings of the 8th IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (EUC '10)*, pp. 200–207, Hong Kong, China, December 2010.

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

[13] K. Akkaya, F. Senel, A. Thimmapuram, and S. Uludag, "Distributed recovery from network partitioning in movable sensor/actor networks via controlled mobility," *IEEE Transactions on Computers*, vol. 59, no. 2, pp. 258–271, 2010.

[14] M. Younis, S. Lee, S. Gupta, and K. Fisher, "A localized self-healing algorithm for networks of moveable sensor nodes," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '08)*, pp. 1–5, December 2008.

[15] M. Imran, M. Younis, A. M. Said, and H. Hasbullah, "Volunteer-instigated connectivity restoration algorithm for wireless sensor and actor networks," in *Proceedings of the IEEE International Conference on Wireless Communications, Networking and Information Security (WCNIS '10)*, Beijing, China, June 2010.