World Applied Sciences Journal 13 (4): 909-915, 2011
ISSN 1818-4952
© IDOSI Publications, 2011

Self Healing by Clustered in Wireless Sensor Networks

[1]*Abolfazl Akbari,* [2]*Amir Panah,* [3]*Omid Panah and* [4]*Neda Beik Mahdavi*

[1,3]Department of Computer Engineering, Islamic Azad University Ayatollah Amoli Branch, Amol, Iran
[2]Young Researchers Club, Qazvin Branch, Islamic Azad University, Qazvin, Iran
[4]Young Researchers Club, Ayatollah Amoli Branch, Islamic Azad University, Amol, Iran

Abstract: Recent increasing growth of interest in Wireless Sensor Networks (WSNs) has provided us a new paradigm to design wireless environmental monitoring applications in the 21st century. However, the nature of these applications and network operational environment has also put strong impact on sensor network systems to maintain high service quality. One challenge is to design efficient fault management solutions to recover network systems from various unexpected failures. In this paper, we design techniques to maintain the cluster structure in the event of failures caused by energy-drained nodes. Initially, node with the maximum residual energy in a cluster becomes cluster heed and node with the second maximum residual energy becomes secondary cluster heed. Later on, selection of cluster heed and secondary cluster heed will be based on available residual energy. We use Matlab software as simulation platform quantities. Like, energy consumption at cluster and number of clusters is computed in evaluation of proposed algorithm. Eventually we evaluated and compare this proposed method against previous method and we demonstrate our model is better optimization than other method, in energy consumption rate.

Key words: Energy efficient · Sensor networks · Transmission range

INTRODUCTION

Technologies of miniaturizing the size of low-cost electronic appliances have boosted the visions of deploying large-scale and dense sensor nodes to extract useful information from harsh environments. However, such small dimension design also puts strong restrictions on the hardware and software capability of a sensor node, in terms of its processing capability, memory storage and energy supply. Among them, the limited power supply is one of the most important constraints. The amount of energy consumed in a radio transmission is proportional to the square of the transmission range. Since the distance from sensor node to sensor node is shorter than from sensor node to the base station, it is not energy efficient for all sensor nodes to send their data directly to a distant base station. Therefore cluster-based data gathering mechanisms effectively save energy. In clustered networks, it creates holes in the network topology and disconnects the clusters, thereby causing data loss and connectivity loss. Good *numbers* of fault tolerance solutions are available but they are limited at different levels. Existing approaches are based on hardware faults and consider hardware components Malfunctioning only. Some assume that system software are already fault tolerant as in [1, 2]. Some are solely focused on fault detection and do not provide any recovery mechanism [3].

Therefore, it is important to identify failed nodes to guarantee network connectivity and avoid network partitioning. The New Algorithm recovery scheme is compared to Venkataraman algorithm proposed in [4]. The Venkataraman algorithm is the latest approach towards fault detection and recovery in wireless sensor networks and proven to be more efficient than some existing related work. It solely focused on nodes notifying its neighboring nodes to initiate the recovery mechanism. It can be observed from the simulation results that failure detection and recovery in our proposed algorithm is more energy efficient and quicker than that of Venkataraman algorithm. In [5], it has been found that Venkataraman algorithm is more energy efficient in comparison with Gupta and Younis [6].

Therefore, we conclude that our proposed algorithm is also more efficient than Gupta and Crash fault detection algorithm in term of fault detection and recovery.

Corresponding Author: Abolfazl Akbari, Department of Computer Engineering,
Islamic Azad University Ayatollah Amoli Branch, Amol, Iran.

This paper is organized as follows: Section 2 provides a brief review of related work in the literature. In section 3, we provided a detail description of our clustering algorithm.

In section 4, we provided a detail description of our proposed solution. The performance evaluation of our proposed algorithm can be found in Section 5, Finally, section 6 concludes the paper.

Related Work: In this section we will give an overview about existing fault detection and recovery approaches in wireless sensor networks. A survey on fault tolerance in wireless sensor networks can be found in [7]. A detailed description on fault detection and recovery is available at [8]. WinMS [9] provides a centralized fault management approach. It uses the central manager with global view of the network to continually analyses network states and executes corrective and preventive management actions according to management policies predefined by human managers. The central manager detects and localized fault by analyzing anomalies in sensor network models. The central manager analyses the collected topology map and the energy map information to detect faults and link qualities. WinMS is a centralized approach and approach is suitable for certain application. However, it is composed of various limitations. It is not scalable and cannot be used for large networks. Also, due to centralize mechanism all the traffic is directed to and from the central point. This creates communication overhead and quick energy depletions. Neighboring co-ordination is another approach to detect faulty nodes. For Example, the algorithm proposed for faulty sensor identification in [10] is based on neighboring co-ordination. In this scheme, the reading of a sensor is compared with its neighboring' median reading, if the resulting difference is large or large but negative then the sensor is very likely to be faulty. Chihfan *et al.* [11] developed a Self monitoring fault detection model on the bases of accuracy. This scheme does not support network dynamics and required to be pre configured. In [12], a fault tolerance management architecture has been proposed called MANNA (Management architecture for wireless sensor networks). This approach is used for fault diagnosis using management architecture, termed as MANNA. This scheme creates a manager located externally to the wireless sensor network and has a global vision of the network and can perform complex operations that would not be possible inside the network. However, this scheme performs centralized diagnosis and requires an external manager. Also, the communication between nodes and the manager is too expensive for WSNs. In Crash fault detection scheme [13], an initiator starts fault detection mechanism by gathering information of its neighbors to access the neighborhood and this process continue until all the faulty nodes are identified. Gathering neighboring nodes information consumes significant energy and time consuming. It does not perform recovery in terms of failure. Gupta algorithm [6] proposed a method to recover from a gateway fault. It incorporates two types of nodes: gateway nodes which are less energy constrained nodes (cluster headers) and sensor nodes which are energy constrained. The less energy constrained gateway nodes maintain the state of sensors as well as multi-hop route for collecting sensors. The disadvantage is that since the gateway nodes are less energy constraint and static than the rest of the network nodes and they are also fixed for the life of the network. Therefore sensor nodes close to the gateway node die quickly while creating holes near gateway nodes and decrease network connectivity.

Also, when a gateway node die, the cluster is dissolved and all its nodes are reallocated to other healthy gateways. This consume more time as all the cluster members are involved in the recovery process. Venkataraman algorithm [5], proposed a failure detection and recovery mechanism due to energy exhaustion. It focused on node notifying its neighboring nodes before it completely shut down due to energy exhaustion. They proposed four types of failure mechanism depending on the type of node in the cluster. The nodes in the cluster are classified into four types, boundary node, pre-boundary node, internal node and the cluster head. Boundary nodes do not require any recovery but pre-boundary node, internal node and the cluster head have to take appropriate actions to connect the cluster. Usually, if node energy becomes below a threshold value, it will send a fail_report_msg to its parent and children. This will initiate the failure recovery procedure so that failing node parent and children remain connected to the cluster.

Cluster Formation: The sensor nodes are dispersed over a terrain and are assumed to be active nodes during clustering.

Problem Definition: The clustering strategy limits the admissible degree, *D* and the number of nodes in each cluster, *S*. The clustering aims to associate every node with one cluster. Every node does not violate the admissible degree constraint, *D* and every cluster does not violate the size constraint, *S* while forming the cluster.

The number of clusters(*C*) in the network is restricted to a minimum of N/S, N > C > N/S, where *N* is the number of nodes in the terrain.

Sensor Network Model: A set of sensors are deployed in a square terrain. The nodes possesses the following properties

- The sensor nodes are stationary.
- The sensor nodes have a sensing range and a transmission range. The sensing range can be related to the transmission range, Rt> 2rs.
- Two nodes communicate with each other directly if they are within the transmission range.
- The sensor nodes are assumed to be homogeneous i.e. they have the same processing power and initial energy.
- The sensor nodes are assumed to use different power levels to communicate within and across clusters.
- The sensor nodes are assumed to know their location and the limits *S* and *D*.

Description of the Clustering Algorithm: Initially a set of sensor nodes are dispersed in the terrain. We assume that sensor nodes know their location and the limits *S* and *D*. Algorithms for estimating geographic or logical coordinates have been explored at length in the sensor network research [14].

In our algorithm, the first step is to calculate *Eth* and *Eic* for every node i, N >I >1. Eth is the energy spent to communicate with the farthest next hop neighbor. *Eic* is the total energy spent on each link of its next hop neighbors. Every node *i* has an initial energy, *Einit*. A flag bit called *"covered flag"* is used to denote whether the node is a member of any cluster or not. It is set to 0 for each node initially.

Calculation of Eth and Eic:

- Nodes send a message *hello_msg* along with their coordinates which are received by nodes within the transmission range. For example in Fig. (1) nodes *a,b,c,d,w,x,y* are within transmission range of *v*.
- After receiving the *hello_msg*, the node *v* calculates the distance between itself and nodes *a, b, c,d,w,x,y* using the coordinates from *hello_msg*. It stores the distance *di* and the locations in the *dist_table*.
- Nodes within the sensing range are the neighbors of a node. In Fig. (1) nodes *w,x,y,b* are neighbors of *v*.

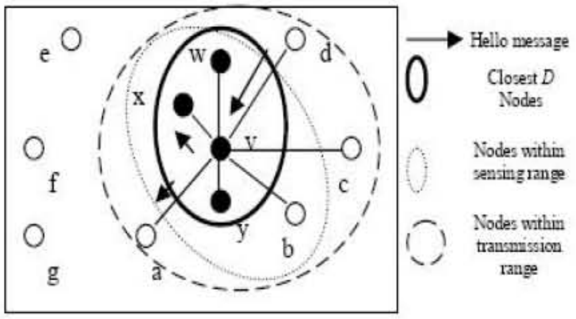

Fig. 1: Topology

- Among the nodes within the sensing range, it chooses the first *D* closest neighbors as its potential candidates for next hop. Assuming *D*=3, in Fig. (1), the closest neighbors of *v* are *w,x,y*.
- Among the potential candidates, the farthest node's distance, *dmax* is taken for the calculation of *E th*.
- Suppose a node needs power E to transmit a message to another node who is at a distance 'd' away, we use the formula E = E=kd ^{c} [7,8], where k and c are constants for a specific wireless system. Usually 2<c<4. In our algorithm we assume k=1, c=2. For a node v, d *1*d^2max = E th_v, since there are D members to which a node sends message.
- E_{ic} is the total energy spent on each of link of the *D* closest neighbors. For a node *v*,

$$E_{icv} = \sum_{i=1}^{D} d_{iv}^{2}$$

Where k= 1. *div* is the distance between node *i* and node *v*. After the calculation of threshold energy *Eth*, nodes become eligible for cluster head position based on their energies. A node *v* becomes eligible for the cluster head position if its *Einit> Ethv. and.* node with the second *Einit> Ethv* becomes secondary Cluster heed. When no nodes satisfy this condition or when there is insufficient number of cluster heads, the admissible degree *D* is reduced by one and then *Eth* is recalculated. The lowest value that *D* can reach is one. In a case where the condition *Einit> Ethv* is never satisfied at all, clustering is not possible because no node can support nodes other than itself. There may also be situations where all the nodes or more number of nodes are eligible for being cluster heads. A method has been devised by which the excess cluster heads are made to relinquish their position.

Relinquishing of the Cluster Head Position

- Every cluster head sends a message *cluster_head_status msg* and *Eic* to its neighbors (within sensing range).
- Every cluster head keeps a list of its neighbor cluster heads along with its *Eic*.
- The nodes which receive *Eic* lesser than itself relinquishes its position as a cluster head.
- The cluster heads which are active send their messages to the cluster_head_manager outside the network. The cluster_head_manager has the information of the desired cluster head count.
- If the number of cluster heads are still much more than expected, then another round of cluster head relinquishing starts. This time the area covered would be greater than sensing range.
- The area covered for cluster head relinquishing keeps increasing till the desired count is reached.

Choosing Cluster Members:

- The cluster head select the closest *D* neighbours as next hop and sends them the message *cluster_join_msg*. The *cluster_join_msg* consists of cluster ID, *Sa*, *D*, *S*, *covered flag*. *Sa* is (*S*-1)/number of next hop members.
- Energy is expended when messages are sent. This energy, *Eic* is calculated and reduced from the cluster head's energy.
- The cluster head's residual energy $Er = Einit - Eic$. *Einit* is the initial energy when the cluster is formed by the cluster head.
- After receiving the *cluster_join_msg*, the nodes send a message, *cluster_join_confirm_msg* to the cluster head if they are uncovered, else they send a message, *cluster_join_reject_msg*.
- After a *cluster_join_confirm_msg*, they set their *covered_flag* to 1.
- The next hop nodes now select *D*-1 members as their next hop members. *D*-1 members are selected because they are already associated with the node which selected them. For example, in Fig. (1) where *D* =3, a node *v* selects node *w*, node *x* and node *y*. In the next stage, node y selects only node *a* and node *b* because it is already connected to node *v* making the *D* = 3.
- After selecting the next hop members, the residual energy is calculated, $Er(\text{new}) = Er(\text{old}) - Eic$
- This proceeds until S is reached or until all nodes have their *covered_flag* set to 1.

Tracking of the Size: The size *S* of the cluster is tracked by each and every node. The cluster head accounts for itself and equally distributes *S*-1 among its next hop neighbors by sending a message to each one of them. The neighbors that receive the message account for themselves and distribute the remaining among all their neighbors except the parent. The messages propagate until they reach a stage where the size is exhausted. If the size is not satisfied, then the algorithms terminates if all the nodes have been covered. After the cluster formation, the cluster is ready for operation. The nodes communicate with each other for the period of network operation time.

Cluster Heed Failure Recovery Algorithm: We employ a back up secondary cluster heed which will replace the cluster heed in case of failure. no further messages are required to send to other cluster members to inform them about the new cluster heed. Cluster heed and secondary cluster heed are known to their cluster members. If cluster heed energy drops below the threshold value, it then sends a message to its cluster member including secondary cluster heed. Which is an indication for secondary cluster heed to stand-up as a new cluster heed and the existing cell manager becomes common node and goes to a low computational mode. Common nodes will automatically start treating the secondary cluster heed as their new cluster heed and the new cluster heed upon receiving updates from its cluster members; choose a new secondary cluster heed. Recovery from cluster heed failure involved in invoking a backup node to stand-up as a new cluster heed.

Performance Evaluation: The energy model used is a simple model shown in [4] for the radio hardware energy dissipation where the transmitter dissipates energy to run the radio electronics and the power amplifier and the receiver dissipates energy to run the radio electronics. In the simple radio model [4], the radio dissipates Eelec = 50 nJ/bit to run the transmitter or receiver circuitry and Eamp = 100 (pJ/bit)/m^2 for the transmit amplifier to achieve an acceptable signal-to-noise ratio. We use MATLAB Software as the simulation platform, a high performance discrete-event Java-based simulation engine that runs over a standard Java virtual machine.

Table 1: Simulation parameters

Simulation parameters	Value
Terrain dimensions	1km^2
Total number of nodes in terrain, *N*	100-1200
Transmissuion range	100-450 m
Cluster size limit, *S*	10-50
Supportable degree, *D*	3-10

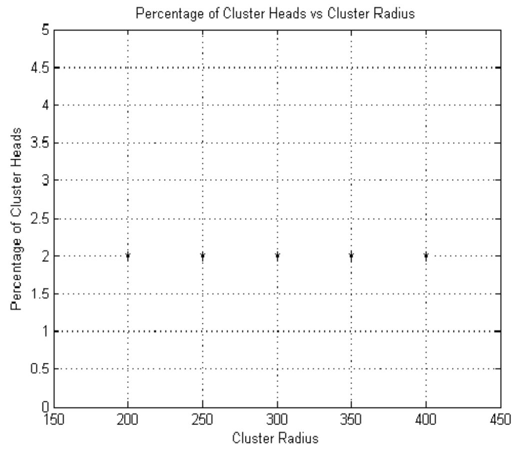

Fig. 2: Percentage of cluster heads with varying cluster radius

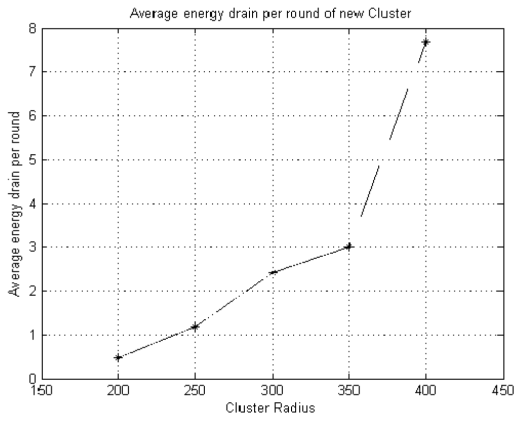

Fig. 3: Ratio of average balance energy drain per round with varying cluster radius

The simulation parameters are explained in Table 1.

Characteristics of the Clusters: Fig. 1 depicts the percentage of cluster heads observed with varying cluster ranges. The cluster range was varied from 200 to 400. The size limit, S in our algorithm was set to 50 with admissible degree, D set to 3. The percentage of cluster heads was observed and noted for about 10 runs of the clustering algorithm. The percentage of cluster heads does not vary over various runs of the algorithm. This is because for a total number of N nodes in terrain, the limit S is set to 50 leading to N/50 cluster heads or clusters. Due to this limitation the results do not show a decrease or an increase in the cluster heads. Though the percentage of cluster heads is not changing, the role of cluster heads is exchanged among the nodes in the network.

Energy Characteristics in Clusters: Fig. 3 depict the energy drain during the cluster formation. Energy drain refers to the loss of energy in each node after every round of cluster formation and operation. The energy loss is according to the relation in the first order radio model. Total energy loss would be the energy loss due to transmission added to the loss due to receiving. energy consumption depends on parameters used in first order radio model, distance and the number of bits, k. This energy consumption is also dependent on how many nodes would the concerned node be transmitting to and receiving from. In this clustering algorithm, the distance is sensing range, which is about 50 % of the transmission range. Also the number of nodes each node would handle is D. These two factors make the loss of energy at every stage uniform.

Evaluation New Algorithm: We compared our work with that of Venkataraman algorithm [5], which is based on recovery due to energy exhaustion. In Venkataraman algorithm, nodes in the cluster are classified into four types: boundary node, pre-boundary node, internal node and the cluster head. Boundary nodes does not require any recovery but pre-boundary node, internal node and the cluster head have to take appropriate actions to connect the cluster. Usually, if node energy becomes below a threshold value, it will send a fail_report_msg to its parent and children. This will initiate the failure recovery procedure so that failing node parent and children remain connected to the cluster. A join_request_mesg is sent by the healthy child of the failing node to its neighbors. All the neighbors with in the transmission range respond with a join_reply_mesg/join_reject_mesg messages. The healthy child of the failing node then selects a suitable parent by checking whether the neighbor is not one among the children of the failing node and whether the neighbor is also not a failing node. In our proposed mechanism, common nodes does not require any recovery but goes to low computational mode after informing their cell managers. In Venkataraman algorithm, cluster head failure cause its children to exchange energy messages.

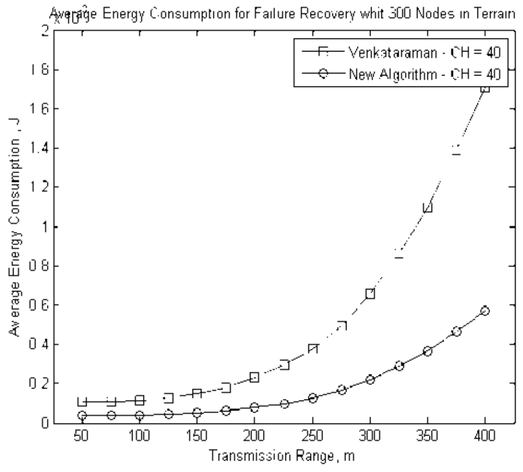

Fig. 4: Average energy loss for cluster-head recovery

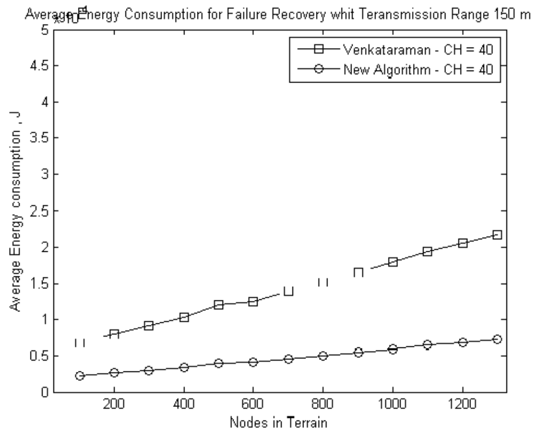

Fig. 5: Average energy loss for cluster-head recovery

The children who are failing are not considered for the new cluster-head election. The healthy child with the maximum residual energy is selected as the new cluster head and sends a final_CH_mesg to its members. After the new cluster head is selected, the other children of the failing cluster head are attached to the new cluster head and the new cluster head becomes the parent for these children. This cluster head failure recovery procedure consumes more energy as it exchange energy messages to select the new cluster head. Also, if the child of the failing cluster head node is failing as well, then it also require appropriate steps to get connected to the cluster. This can abrupt network operation and is time consuming. In our proposed algorithm, we employ a back up secondary cluster heed which will replace the cluster heed in case of failure no further messages are required to send to other cluster members to inform them about the new cluster heed Figs. 4 and 5 compare the average energy loss for the failure recovery of the three algorithms. It can be observed from Fig. 4 that when the transmission range increases, the greedy algorithm expends the maximum energy when compared with the Gupta algorithm and the proposed algorithm. However, in Fig. 5, it can be observed that the Gupta algorithm spends the maximum energy among the other algorithms when the number of nodes in the terrain increases.

CONCLUSION

In this paper, we have proposed a Self Healing algorithm, which is energy-efficient and responsive to network topology changes due to sensor node failures. The proposed cluster-head failure-recovery mechanism recovers the connectivity of the cluster in almost less than of the time taken by the fault-tolerant clustering proposed by Venkataraman. The Venkataraman algorithm is the latest approach towards fault detection and recovery in wireless sensor networks and proven to be more efficient than some existing related work. Venkataraman algorithm is more energy efficient in comparison with Gupta and Algorithm Greedy Therefore, we conclude that our proposed algorithm is also more efficient than Gupta and Greedy [15] algorithm in term of fault recovery. The faster response time of our algorithm ensures uninterrupted operation of the sensor networks and the energy efficiency contributes to a healthy lifetime for the prolonged operation of the sensor network.

REFERENCES

1. Koushanfar, F., M. Potkonjak and A. Sangiovanni-Vincentelli, 2002. Fault Tolerance in Wireless Ad-hoc Sensor Networks, Proceedings of IEEE Sensors.
2. Lee, W.L., A. Datta and R. Cardell-Oliver, 0000. Network Management in Wireless Sensor Networks, to appear in Handbook on Mobile Ad Hoc and Pervasive Communications, edited by M.K. Denko and L.T. Yang, American Scientific Publishers.
3. Paradis, L. and Q. Han, 2007. A Survey of Fault Management in Wireless Sensor Networks, J. Network and Systems Management, 15(2): 171 190.
4. Venkataraman, G., S. Emmanuel and S. Thambipillai, 16. Gupta, G. And M. Younis, 2003. 2008. Energy-efficient cluster-based scheme for 'Fault-tolerant clustering of wireless sensor failure management in sensor networks IET networks'. Proc. IEEE WCNC, New Orleans, Commun, 2(4): 528-537.

5. Venkataraman, G., S. Emmanuel and S. Thambipillai, 2008. Energy-efficient cluster-based scheme for failure management in sensor networks IET Commun, 2(4): 528-537.
6. Meng, X., T. Nandagopal, E. Li and S. Lu, 2006. November 2003, pp: 315-324. Contour Maps: Monitoring and Diagnosis in Sensor.
7. Gupta, G. and M. Younis, 0000. Fault tolerant Sensor Networks, Lisboa, Portugal, July 2006. clustering of wireless sensor networks; WCNC'03,
8. Chessa, S. and P. Santi, 2002. Crash faults 23. Y. Mei, C. Xian, S. Das, Y.C. Hu and Y.H. Lu, 2006. identification in wireless sensor networks, Comput. Repairing sensor networks using mobile robots. Proc. Commun., 25(14): 1273-1282.
9. Zhou, Z., S. Das and H. Gupta, 2005. Fault tolerant detection in wireless sensor network, IEEE Trans. connected sensor cover with variable sensing and Comput., 53(3): 241-250. Transmission Ranges. Proc. IEEE Sensor and Ad Hoc.
10. Harte, S., A. Rahman and K.M. Razeeb, 2005. Fault 28. Ganeriwal, S., A. Kansal and M.B. Srivastava, 2004. tolerance in sensor networks using self diagnosing Self aware actuation for fault repair in sensor sensor nodes. Proc. IEE Int. Workshop on Intelligent networks. Proc. IEEE Int. Conf. Robotics and Environments, UK, June 2005, pp: 7-12. Automation, New Orleans, USA, April 2004.
11. Oates, T., 1995. Fault Identification in Computer, 5: 5244-5249. network: a review and a new approach. Technical.1
12. Ding, M., D. Chen, K. Xing and X. Cheng, 2005. Networks, Nice, France, May 2006, 138(23). Localized fault-tolerant event boundary detection in USA., March 2005, 2: 902-913. IEEE Int. Conf. Network Protocols, Atlanta, USA.
13. Staddon, J., D. Balfanz and G. Durfee, 2002. Efficient 4: 2302-2312. tracing of failed nodes in sensor networks. Proc. 1st.
14. Ruiz, L.B., I.G. Siqueira, L.B. Oliveira, H.C. Wong, Italy, September 2005, pp: 508-512. J.M.S. Nogueira and A.A.F. Loureiro, 2004. Fault.
15. Perrig, A., R. Szewczyk, V. Wen and D.E. Culler, 2005. DASCA: a degree and size based clustering J.D. Tygar, 2002. SPINS: security protocols for approach for wireless sensor networks. Proc. IEEE, sensor networks, Proc. Wirel. Netw., 8(5): 521-534. Int. Symp. Wireless Communication Systems, Siena.