A FAST FAULT TOLERANT PARTITIONING ALGORITHM FOR WIRELESS SENSOR NETWORKS

Dibakar Saha[1] and Nabanita Das[2]

Advanced Computing and Microelectronics Unit,
Indian Statistical Institute Kolkata, India
[1]dibakar.saha10@gmail.com, [2]ndas@isical.ac.in

ABSTRACT

In this paper, given a random uniform distribution of sensor nodes on a 2-D plane, a fast self-organized distributed algorithm is proposed to find the maximum number of partitions of the nodes such that each partition is connected and covers the area to be monitored. Each connected partition remains active in a round robin fashion to cover the query region individually. In case of a node failure the proposed distributed fault recovery algorithm reconstructs the affected partition locally utilizing the available free nodes. Simulation studies show significant improvement in performance compared to the earlier works in terms of computation time, the topology of each partition, message overhead and network lifetime.

KEYWORDS

Wireless Sensor Network (WSN), Self Organization, Coverage, Partition, Network Lifetime, Fault Resilience

1. INTRODUCTION

In a Wireless Sensor Network (WSN), a large number of sensor nodes are randomly deployed to monitor a large geographical area. Each sensor node is integrated with processing elements, memory, battery power and wireless communication capabilities. Once deployed, they are, in general left unattended. Due to power drainage, hardware degradation, environmental hazards etc. sensor nodes may fail. To combat frequent failures of non-repairable nodes, the networks are generally over-deployed. For better utilization of the over-deployed nodes to save energy and to extend the lifetime of the network, this paper addresses the problem of finding the maximum number of partitions of the sensor nodes such that each partition is connected and covers the whole query region. Instead of keeping all the sensor nodes active always, these partitions will remain active one after another in a round robin fashion. Therefore, if there exists K such partitions, the network lifetime can be enhanced by at most K times. Here, given a random uniform distribution of sensor nodes over a 2-D plane, a fast distributed algorithm is developed for finding the maximum number of partitions of connected nodes such that each partition ensures coverage. In case of node failures, a distributed algorithm is developed for fault recovery that rearranges the affected partition locally to tolerate single node faults within a partition. Simulation studies show that compared to the earlier techniques, the proposed algorithm is faster and results better partition topology with reduced diameter and requires less message overhead. Also, in case of unpredictable node faults the neighboring nodes in the same partition execute the

Sundarapandian et al. (Eds) : ACITY, AIAA, CNSA, DPPR, NeCoM, WeST, DMS, P2PTM, VLSI - 2013
pp. 227–237, 2013. © CS & IT-CSCP 2013 DOI : 10.5121/csit.2013.3424

localized fault recovery algorithm that rearranges the partition locally to make the system fault resilient. Simulation results show that it extends the network lifetime significantly.

The rest of the paper is organized as follows. Section 2 presents a brief outline of related works. Section 3 describes the proposed model. Section 4 presents the proposed algorithms. Simulation results are included in Section 5 and Section 6 concludes the paper.

2. RELATED WORKS

Extensive research results have been reported so far addressing the problems of sensing coverage and network connectivity in wireless sensor networks. In many works, the authors considered only the coverage issue in wireless sensor networks [1]–[6]. But unless the coverage and connectedness problems are considered jointly, the data sensed by the nodes covering the region cannot be gathered at the sink node in multi-hop WSN's. The problem of finding a connected set cover of minimum size is itself an NP-hard problem [7]. Authors of [7], [8] focused on both connectivity and coverage problems with the objective of finding a single connected set cover only. But finding just a single connected cover keeps a large number of sensors unutilized. Some of the papers considered the fault tolerant connected set cover problem. An approximation algorithm is proposed in [9] for fault tolerant connected set cover problem. In [10], a coverage optimization algorithm based on particle swarm optimization technique is developed. In [11]–[14], authors proposed several dynamic localized algorithms to achieve the coverage and connectivity. But dynamic algorithms, in general, require substantial message overhead to collect recent neighborhood information at regular intervals. Hence, the authors in [15], propose a localized algorithm for finding maximum number of connected set covers that is to be computed just once during network initialization only.

In some papers [12], [14], [15], it has been assumed that the query area is a dense grid [16], [17] composed of unit cells. The knowledge of exact location of each node is needed here. A sensor node computes the covered area by counting the cells covered by each neighbor that makes the procedure computation intensive. To avoid this, in [18] the DCSP algorithm is proposed where authors assume that the monitoring area is divided into a limited number of square blocks such that a sensor node within a block completely covers it irrespective of its position within the block. Therefore, the coverage problem can be solved easily with much less computation. However, the proposed distributed algorithm was a slow one requiring p rounds to achieve a partition with p nodes. Also, the fault model considered the faults due to energy exhaustion only where a node can predict its failure and can inform its neighbors in advance. In this paper, a faster distributed algorithm requiring less message overhead is proposed that is executed during network initialization only. It attempts to create maximum number of connected partitions of sensor nodes with reduced diameter such that each partition covers the area under investigation and being active in round robin fashion it enhances the network lifetime manifold. The reduced diameter of the partition keeps the communication latency low. Moreover, a distributed fault recovery algorithm is developed for a stronger fault model that in presence of any unpredictable node faults, can rearrange the affected partition locally, so that it remains operational. Simulation studies show that this fault recovery scheme enhances the network lifetime by more than 50%.

3. PROPOSED MODEL AND PROBLEM FORMULATION

Let n homogeneous sensor nodes be deployed over a 2-D region P each with sensing range S and transmission range T. It is assumed that P is divided into a finite number of square blocks [18]. Each side of the block is $R/\sqrt{2}$ as shown in Figure1, where $R=min(S, T)$. Therefore, it is evident that each sensor node within a block B completely covers B and all nodes within the same block are always connected to each other.

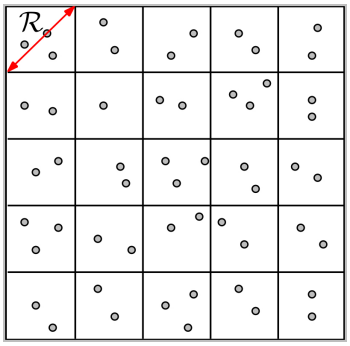


Figure 1. Region *P* divided into a square blocks with diagonal *R*

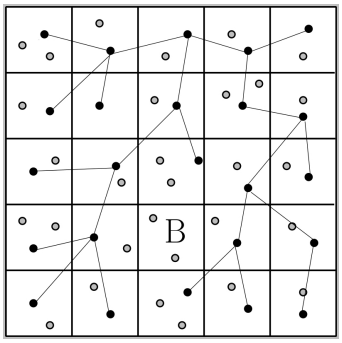


Figure 2. A Connected partition of nodes

Hence, activating just a single sensor node from each block is sufficient to cover the region P. But it is not guaranteed that any such set is connected.

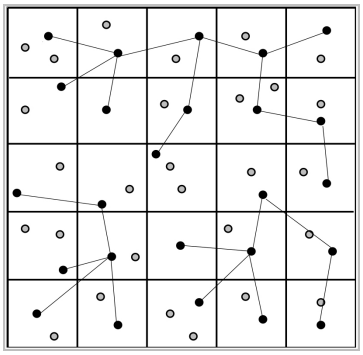


Figure 3. A partition that covers *P* but is disconnected

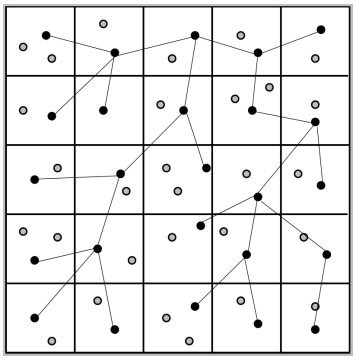


Figure 4. A partition with coverage and connectivity

As for example, Figure *2* shows a partition where the selected nodes (shown by solid circles) are connected but a block *B* is left uncovered. Whereas, Figure *3* shows a partition where all blocks are covered but the selected nodes are not connected, and finally, Figure *4* shows the desired topology where the partition covers all blocks as well as it is connected. Assuming this grid structure of the query region *P* , this paper addresses the connected set cover partitioning problem introduced in [15]. For completeness, the problem is defined below.

Definition 1. *Consider a sensor network consisting of a set S of n sensors and a query region P . A set of sensors $N \subseteq S$ is said to be a connected K-cover for P if, each point $p \in P$ is covered by at least K sensors from N and the communication graph induced by N is connected.*

a) **Connected Set Cover Problem:** *Given n sensor nodes distributed over a query region, the Connected Set Cover Problem is to find a connected 1-cover of minimum size. This problem is known to be an NP-hard problem [7].*

b) **Connected Set Cover Partitioning Problem***: The Connected Set Cover Partitioning Problem is to partition the sensors into connected 1-covers such that the number of covers is maximized [15].*

It is evident that the problem (b) is at least as hard as problem (a). The following section describes the algorithms developed for solving the Connected Set Cover Partitioning Problem.

4. ALGORITHM FOR CONNECTED SET COVER PARTITIONING

In pervasive computing environments, it is evident that in most of the cases the system captures data in distributed nodes communicating through poorly connected network. Since, in WSN large number of sensor nodes is deployed over a geographical area, to collect information of the whole network at a central node is not feasible. Instead, it is better to compute in a distributed fashion based on the local neighborhood information using less communication. Hence the focus of our work is on distributed computation of the connected partitions. In this section, a distributed algorithm is developed to find the maximum number of connected-1 covers of a WSN. Also, in case of node fault, a distributed algorithm is presented to rearrange the affected partition locally to make the system fault resilient.

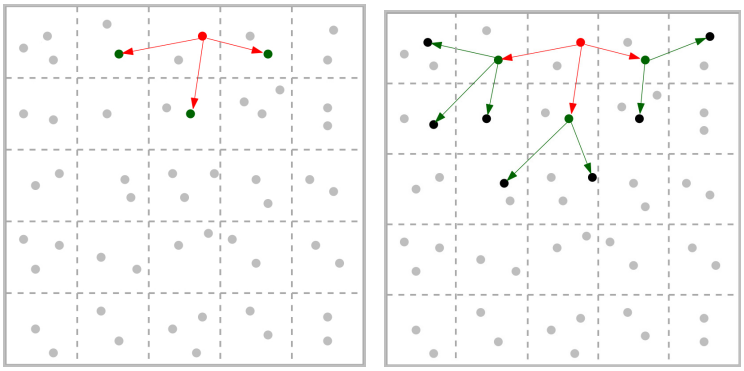
4.1. Distributed Algorithm for Partitioning

It is assumed that a set of n sensor nodes $S=\{s_1, s_2, s_3, ..., s_n\}$ is deployed on a 2-D plane P divided into say, m square blocks $P= \{ p_1, p_2, p_3,...., p_m\}$, each with side $R/\sqrt{2}$ where $R= min(S,T)$, as has been described in Section 3. Each block has unique id. Each sensor node knows its location in terms of its block within which it is located. After deployment, by the neighborhood discovery phase, each node knows its neighbor-id's, their degrees and block-id's respectively.

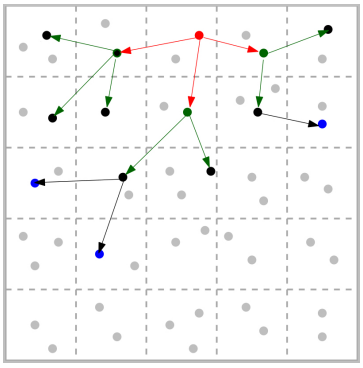
To generate the partitions the following types of messages are exchanged among nodes.

- *Selectlist* (C_i, i, $\{j\}$) : This message is sent by a node- i that selects a list of neighbors $\{j\}$ for inclusion in its partition C_i.
- *Selected* (C_i, $\{j\}$) : This message is initiated by the leader node l_i of C_i and is sent to nodes $\{j\}$ for inclusion in its partition.
- *Confirm* (C_i, j) : A node $\in \{j\}$ sends this message to the leader after joining its partition C_i.
- *Include* (Ci, j) : The leader broadcasts this message within Ci to include node $j \in C_i$.

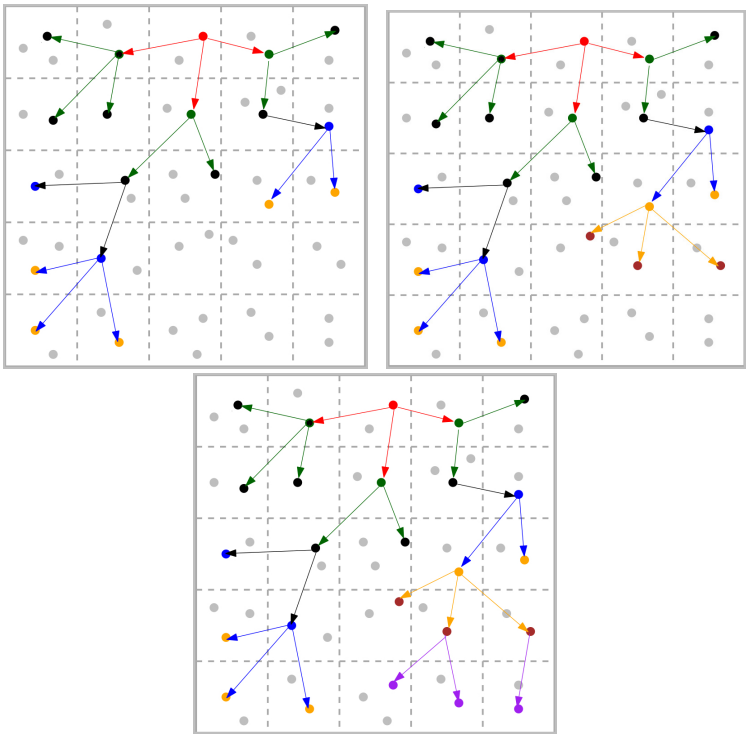
Depending on the node density, a probability value $0 < l_{prob} < 1$ is determined to select p number of leader nodes randomly. Each node i in S generates a random number r to check if $r \leq l_{prob}$, the *leader probability*. If yes, it becomes a leader node and sets its parent as null. Say, p leader nodes emerge and $L =\{ l_1, l_2, l_3, ... , lp\}$. Next each leader node $l_i \in L$ initiates the creation of one partition C_i concurrently to generate a disjoint *connected 1-cover*.



(a) In Round 1 (b) In Round 2



(c) In Round 3



(d) In Round 4 (e) In Round 5 (f) In Round 6

Figure 5. steps for constructing connected 1-cover in successive rounds

In round *1*, each leader l_i in L initiates a partition $C_i=\{l_i\}$. In each round, each node $i \in C_i$ prepares *'Selectlist'* consisting of neighbors of itself and its children in C_i, each one from an uncovered block with minimum degree D. If it finds none, it includes a neighbor from any block with maximum degree D. In case, a node gets more than one node from the same block, it selects the neighbor with minimum D. Node i sends a *'Selectlist'* message to its parent in C_i if it is a leaf node in C_i. Else, after receiving the *'Selectlist'* messages from all its children in C_i, node $j \in C_i$ sends the *'Selectlist'* message to its parent node if it is not a leader node.

The leader node finally selects the nodes to be included and sends the *'Selected'* message to them. If a node receives *'Selected'* messages from more than one nodes, it selects the parent with

minimum degree D and confirms the request by sending a *'Confirm*' message to the corresponding leader. The leader includes the node in C_i and broadcasts the *'Include'* message to all nodes in C_i. On receiving an *'Include(C_i, j)'* message, all nodes $k \in C_i$ include node-*j* in its partition and make necessary updates. In each round, this procedure is repeated until either all blocks are covered by a partition C_i, or no neighbors are left for inclusion.

The formal description of the algorithm is given below.

Algorithm 1: Distributed algorithm for connected set cover partitioning

```
Input:  1-hop neighbor list of each node NL(i) with degree D, Block-Id, Block status, Status,
        Leader Probability : lprob
Output: Partition Ci from leader li

  for each node-i do
      if node-i is a leader then Ci ← {i}; parent = ∅; status = 1;
      end
      if Status = 1 and not all blocks covered then
          prepare 'Selectlist' with neighbors from uncovered blocks with minimum D or
              neighbor from covered block with maximum D;
          if received 'Selectlist' messages from all children ∈Ci and Parent ≠ ∅ then
              prepare 'Selectlist' to cover maximum number of uncovered blocks with
                  neighbors having minimum D;
              Send 'Selectlist' to the parent node;
          else
              if 'Selectlist' =∅ then
                  Broadcast success=0 and terminate;
              else
                  Send 'Selected' message to the nodes in 'Selectlist' ;
              end
          end
      end
      if Status = 0 receives 'Selected' message then
          Select that partition where the sender have minimum D and send 'Confirm'
              message to leader;
          Update NL(i), Block status, Status and include in Ci;
      end
      if leader node and receives 'Confirm' message then
          Broadcast 'Include' message to all nodes in Ci and update NL(i) , Block status,
              Status;
          if all blocks are covered then
              Broadcast success=1 and terminate;
          end
      else
          if status=1 and receives 'include' message then
              update NL(i);
          end
      end
  end
```

Correctness Proof: From the algorithm, it is clear that in each round of the procedure, the nodes already existing in a partition include several neighbors in it. As the partition starts with a single leader node, it always remains connected including new neighbors of the nodes covering additional blocks. The process terminates when at least one node from each block has been included i.e. a successful partition always satisfy the condition of connectedness and coverage. In case of a failure, the nodes in the incomplete partition declare them as free nodes, and take part in other partitions or remain as stand by nodes to facilitate the fault recovery phase.

Example 1. In Figure 5, it is shown that in round 1, the leader node (red) selects the neighbors (the green ones) from uncovered blocks. In the next round, all black nodes are selected by the red and green nodes. This procedure is repeated to include brown and blue nodes until all blocks are covered. In the last round, all purple nodes are selected and the process is terminated as no uncovered block exists.

4.2. Distributed Fault Recovery Algorithm

As it has been mentioned in Section 1, once deployed the sensor nodes may fail due to low energy, hardware degradation, inaccurate readings, environmental changes etc. This paper focuses on the fault recovery problem in case of a single unpredictable node fault within a partition. It is assumed that when an active node f of a partition C_i fails abruptly, its parent (if exists) and the children (if exists) in C_i can detect it.

A fast fault recovery algorithm by which all children and the parent of the faulty node in the partition after detecting the fault rearrange the partition quickly to make the partition connected with full coverage. In case, the faulty node is the leader node, its children with the minimum node-id becomes the new leader otherwise the parent node becomes the leader and the fault recovery procedure is initiated by the new leader.

The formal description of the algorithm is given below.

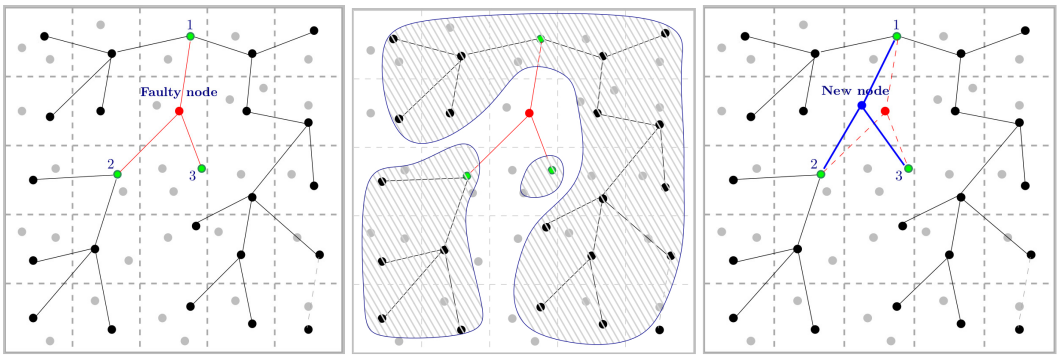
Algorithm 2: Distributed fault recovery algorithm

```
Input: faulty node-id : f, partition: Ci, parent & children of f: Si
Output: Recovered partition Ci

If f is a leader node then leader_temp ← Minimum ID child of f, otherwise
        leader_temp ← parent of f;
        Make a block list B_S[ ] covered by (f ∪ S_i) \ leader_temp with status ← 0 ;
        include leader_temp in temp[ ];
end
for each node-i in temp[ ] do
        find neighbors in S_i or from uncovered block. If none, select the free neighbor
with      maximum D; Include neighbors in 'Selectlist', ;
        if node-i not leader_temp then
                Send 'Selectlist' message to leader_temp;
        else
                if leader_temp receives 'Selectelist' message from all nodes-j in temp[ ]
then
                        Include nodes in temp[ ]; Update B_S[ ];
                        if B_S[ ] == ∅ then
                            Send 'FaultRecovered' message with temp[ ] to ∀j ∈ Ci
and
                                terminate;
                                if 'Selectlist' = ∅ for ∀j ∈ temp[ ] then
                                    Broadcast 'RecoveryFailed' and terminate;
                                else
                                    send temp[ ] and B_S[ ] to all nodes in temp[ ];
                                end
                        end
        end
    end
```

Correctness Proof: In the fault recovery algorithm, in case a node $i \in C_i$ fails, a unique leader node $leader_{temp} \in C_i$ is selected from the parent or children of *i*. $leader_{temp}$ marks the set of blocks

of the remaining children of i as uncovered. Then it initiates the fault recovery procedure, in the same fashion as it has been done in algorithm 1, only with the difference that here the children of the faulty node in the uncovered blocks are preferred than other nodes to be included in C_i. It guarantees that when the procedure is successful, the repaired partition is again connected and covers the whole region.



(a) Fault Detection (b) Disjoint Sub Partition (c) Fault Recovery

Figure 6. Fault Detection and Recovery

Example 2. In Figure 6, say, the red node is faulty. It will be detected by all its children and parent (colored by green). The partition is broken into three disjoint components as shown in Figure 6(b). The leader node selects its neighbor the blue node from the faulty node's block for maintaining the coverage and it is also connected to at least one node from the disjoint components. Therefore, the connectivity is preserved. Now the new partition including the blue node, is ready for monitoring the area.

5. SIMULATION RESULTS AND DISCUSSION

For simulation studies, we have used network simulator NS 2.34 to evaluate the performance of our proposed algorithm. We have compared our results with [18] that show significant improvement on the number of rounds for generating the partitions, the network diameter and the number of transmitted messages per node during the procedure. The sensor nodes are deployed over a grid P which is divided into a number of blocks (2×2), (3×3) to (7×7) respectively. Figure 7 shows the variation of the average number of rounds to complete partitioning with the grid size. Obviously, the number of rounds increases with the number of blocks. However, compared to the *DCSP* algorithm proposed in [18], the present method completes in significantly less number of rounds. Therefore, during initialization, the proposed method will converge faster to achieve the connected covers of the nodes.

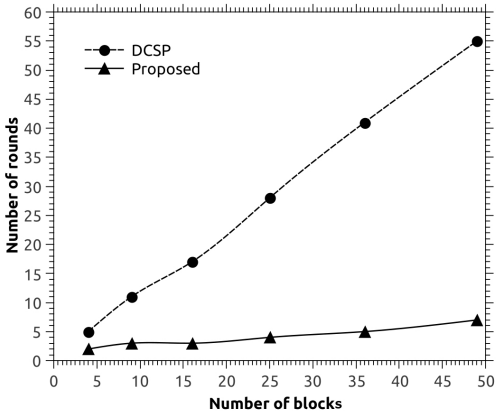


Figure 7. Comparison of average number of rounds for partitioning between *DCSP* and the proposed algorithm

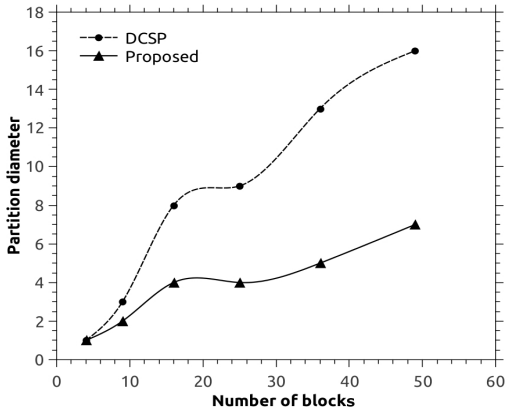


Figure 8. Comparison of diameter between *DCSP* and the proposed algorithm

In figure 8, it is shown that the proposed algorithm results significant improvement in terms of the average network diameter of the generated partition over the DCSP algorithm [18]. In a network with large diameter, the number of steps to route a message from a source node to a destination node will require more delay and more massage exchanges between intermediate nodes. Therefore, the low diameter network topologies are highly preferred for a partition that can aggregate the data using less number of hops, i.e., with less delay and less number of broadcasts.

Figure 9 shows the significant improvement in average number of transmitted messages per node in computing the connected set covers. Since the procedure terminates faster using fewer rounds of computation, the total number of messages exchanged per node is also less here. Therefore the proposed technique is better in terms of energy efficiency also.

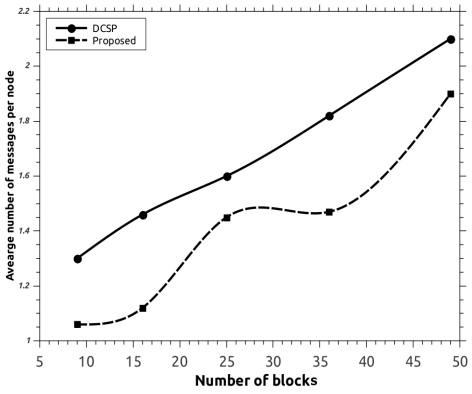


Figure. 9. DCSP vs. Proposed algorithm in terms of average number of transmitted messages

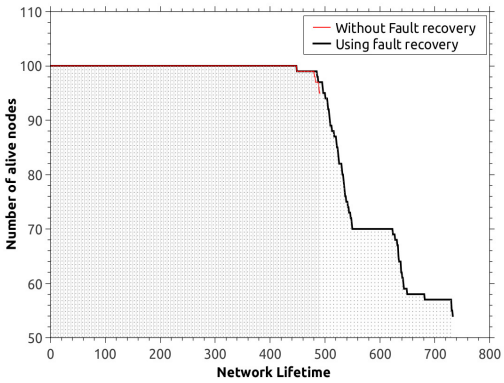


Figure. 10. Extension of network lifetime using fault recovery technique

Finally, Figure 10 shows how the fault recovery algorithm enhances the network lifetime in presence of faults. In the simulation, only node faults due to energy exhaustion has been taken into account. Simulation results show almost 50% enhancements in network lifetime.

6. CONCLUSION

In this paper, we have focused on the connected set cover partitioning problem in Wireless Sensor Networks. A self-organized fast distributed algorithm is proposed for finding maximum number of connected partitions to cover the region. Also, distributed fault recovery technique is developed to rearrange connected set covers in presence of unpredictable node faults to satisfy both connectivity and coverage criteria. Reduction in network diameter of the partition and significant improvements in terms of computation rounds and message overhead are also achieved by the proposed method. In summary, the proposed connected set cover partitioning technique along with the localized fault recovery scheme opens up new avenues for setting up self organized wireless sensor networks with enhanced lifetime.

REFERENCES

[1] X. Li, P. Wan and O. Frieder, "Coverage in Wireless Ad-hoc Sensor Networks," IEEE Transactions on Computers, vol. 52, June 2003, pp. 753–763.

[2] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava, "Coverage problems in wireless ad-hoc sensor networks," in IEEE INFOCOM , 2001, pp. 1380–1387.

[3] D. Wang, B. Xie and D. P. Agrawal, "Coverage and Lifetime Optimization of Wireless Sensor Networks with Gaussian Distribution," IEEE Transactions on Mobile Computing, vol. 7, December 2008, pp. 1444–1458.

[4] J. Jiang and W. Dou, "A Coverage-Preserving Density Control Algorithm for Wireless Sensor Networks," in Ad-Hoc, Mobile, and Wireless Networks, 2004, pp. 42–55.

[5] Huang, Chi-Fu and Tseng, Yu-Chee, "The Coverage Problem in a Wireless Sensor Network," in Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications. ACM, 2003, pp. 115–121.

[6] S. Slijepcevic and M. Potkonjak, "Power efficient organization of wireless sensor networks," in IEEE International Conference on Communications , 2001, pp. 472–476.

[7] H. Gupta, Z. Zhou, S. R. Das and Q. Gu, "Connected sensor cover: self-organization of sensor networks for efficient query execution," IEEE/ACM Transactions on Networking, vol. 14, pp. 55–67, 2006.

[8] Z. Zhou and S. Das and H. Gupta, "Connected K-Coverage Problem in Sensor Networks," in 13th International Conference on Computer Communications and Networks, ICCCN 2004, October 2004, pp. 373– 378.

[9] Z. Zhang, X. Gao and W. Wu, "Algorithms for Connected Set Cover Problem and Fault-Tolerant Connected Set Cover Problem," Theoretical Computer Science, vol. 410, 2009, pp. 812–817.

[10] P. Li, L. Kia and L. Gang, "Research on Wireless Sensor Networks Fault-Tolerant Coverage Algorithm Based on Particle Swarm Optimization," in International Conference on Wireless Sensor Network, IET-WSN, November 2010, pp. 286–290.

[11] C. Lin, C. Chen and A. Chen, "Partitioning Sensors by Node Coverage Grouping in Wireless Sensor Networks," in International Symposium on Parallel and Distributed Processing with Applications (ISPA) , September 2010, pp. 306–312.

[12] D. Tian and N. D. Georganas, "A Coverage-Preserving Node Scheduling Scheme for Large Wireless Sensor Networks," in Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications. ACM Press, 2002, pp. 32–41.

[13] X. Wang, G. Xing, Y. Zhang, C. Lu, Chenyang, R. Pless and C. Gill, "Integrated Coverage and Connectivity Configuration in Wireless Sensor Networks," in Proceedings of the 1st International Conference on Embedded Networked Sensor Systems. ACM, 2003, pp. 28–39.

[14] A. Gallais, J. Carle, D. Symplot-Ryl, and I. Stozmenovic, "Localized sensor area coverage with low communication overhead," IEEE Transactions on Mobile Computing , vol. 7, no. 5, pp. 661–672, 2008

[15] N. Pervin, D. Layek and N. Das, "Localized Algorithm for Connected Set Cover Partitioning in Wireless Sensor Networks," in 1st International Conference on Parallel Distributed and Grid Computing (PDGC). IEEE-xplore, October 2010, pp. 229–234.

[16] W. Ke, B. Liu, M. Tsai, "The Critical-Square-Grid Coverage Problem in Wireless Sensor Networks is NP-Complete," Computer Networks, pp. 2209–2220, February 2010.
[17] R. S. S. Shakkottai and N. Shroff, "Unreliable sensor grids: Coverage, connectivity and diameter," in Proceedings of IEEE INFOCOM, 2003, pp. 1073–1083.
[18] D. Saha and N. Das, "Distributed Area Coverage by Connected Set Cover Partitioning in Wireless Sensor Networks," in 1st International Workshop on Sustainable Monitoring through Cyber-Physical Systems (SuMo-CPS) in Conjunction with ICDCN 2013. ACM Digital Library , January 2013, pp. 17–22.

Authors

Dibakar Saha, received Master of Computer Applications (MCA) degree from University of North Bengal, West Bengal, in 2010. Since 2010 he has been a project researcher at Advanced Computing and Microelectronics unit, Indian Statistical Institute, Kolkata. His research interests include parallel processing and Distributed Computing.

Dr. Nabanita Das is a Professor in the Advanced Computing and Microelectronics Unit of Indian Statistical Institute, Kolkata, India, since 1996. Her area of interest includes Mobile Ad Hoc Networking, Pervasive Computing, Parallel and Distributed Computing and Multi-Core Computing. Dr. Das acted as the co-Editor of 'Distributed Computing- IWDC 2004' LNCS, Springer, Guest Editor of special issue on 'Resource Management in Mobile Communication Networks' of 'Microprocessors and Microsystems', 2004, Elsevier, Editor of 'Recent Trends in High Performance Computing', Proc. of International Workshop on High Performance Computing', 1998 – 99. She serves as the reviewer of many International Journals like IEEE Transactions on Computers, IEEE Transactions on Vehicular Technology, IEEE Transactions on Systems Man and Cybernetics, Journal of Parallel and Distributed Computing etc. She is a senior member of IEEE. She served as the Chair of Women in Engineering (WIE) affinity group of IEEE Calcutta section during 2009-2011.