LOCALIZED MOVEMENT CONTROL FOR FAULT TOLERANCE OF MOBILE ROBOT NETWORKS

Shantanu Das[1], Hai Liu[1], Ajith Kamath[1], Amiya Nayak[1], and Ivan Stojmenović[1,2]

[1] School of Information Technology and Engineering, University of Ottawa, Canada
[2] Electronic, Electrical & Computer Engineering, The University of Birmingham, United Kingdom
{shantdas,akamath,hailiu,anayak,ivan}@site.uottawa.ca

Abstract. In this paper, we present a novel localized movement control algorithm to form a fault-tolerant bi-connected robotic network topology from a connected network, such that total distance of movement of robots is minimized. The proposed distributed algorithm uses p-hop neighbor information to identify critical head robots that can direct two neighbors to move toward each other and bi-connect their neighborhood. Simulation results show that the total distance of movement of robots decreases significantly with our localized algorithm when compared to the globalized one, and our localized algorithm achieved 100% success on considered non-bi-connected networks. To the best of our knowledge, it is the first work on localized movement control for fault tolerance of mobile robot networks.

Keywords : mobile sensors, actor networks, fault tolerance, robot movement, localized control.

1 Introduction

A significant trend in the development of autonomous mobile robot networks is seen due to the advancement of sophisticated robots. Mobile robotics is expected to be a solution ranging from vast number of industrial applications to service robotics. In such applications coordination between individual robots is essentially accomplished through a wireless ad hoc network. For example, coordination of robotic relay stations was studied in [5] to maintain communication between an explorer and a base station. Application of mobile robotics is vast. Potential applications include military missions, unmanned space exploration, and data collection in sensor fields. But for such applications, coordination of a robot team in pursuit of common task is essential. Existing algorithms for mobile robots coordination are suitable for robots with no or very low failure rates. However, when robots are susceptible to failures, as in many applications, it is critical for robotic networks to incorporate the ability to sustain faults and operate normally. Communication faults in robot networks can be caused by hardware damage, energy depletion, harsh environment conditions and malicious attacks. A fault in a robot can cause stopping transmission tasks to others as well as relaying data to sink. Data sent by a robot will be lost if the receiving robot fails. So, a communication link failure on a route requires data to be re-routed. That is, in order to handle general communication faults, there should be at least two node-disjoint paths between each pair of robots in the network. A network is defined to be **bi-connected** if there exist two node-disjointed paths between any pair of nodes in the network, i.e., the removal of any node from the network leaves the network still connected. Therefore, bi-connectivity is the basic requirement for design of fault-tolerant networks [9].

In this paper, we focus on mobile robot networks and study movement control of robots to establish a fault-tolerant bi-connected network. The robot network is assumed to be connected, but not necessarily bi-connected. Achieving connectivity in a disconnected network is difficult due to the lack of communication between the disconnected parts. However, if the network is already connected, we can make it bi-connected (and thus fault-tolerant) by movement of selected robots. Recent work in [2] has shown that fault tolerance can be achieved through globalized robot movement control algorithm. It is a *centralized* algorithm that assumes one of robots or a base station has global information of the network. We focus on the *localized* version of movement control algorithm for building a fault-tolerant robot network. To the best our knowledge, this is the first work on localized movement control for fault tolerance of mobile robot networks.

The rest of the paper is organized as follows. Related work is introduced in Section 2. We propose a localized movement control algorithm to construct bi-connected mobile robot networks in Section 3. Results obtained from extensive simulations are provided in Section 4 to show the effectiveness of our algorithm. Finally we conclude our work in Section 5.

2 Related Work

Many topology control algorithms have been proposed to achieve network reliability in static networks. These algorithms cope with preserving fault tolerance by selecting certain links to neighbors in an already well connected network. The problem of adjusting the transmit power of nodes to create a desired topology in multiple wireless networks was studied in [10]. For static networks, two centralized algorithms were proposed to construct connected and bi-connected networks while minimizing the maximal transmission power of nodes. Two distributed heuristics were further proposed for mobile networks. The basic idea is to adaptively adjust node transmit power according to topological changes and attempt to maintain a connected topology with the minimum power. A more general case for k-vertex connectivity of wireless networks was studied in [7]. Both a centralized algorithm and a localized algorithm were proposed. Both above works assumed that nodes have uniform transmission range. That is, they focused on homogenous networks. Topology control in heterogeneous wireless networks was discussed in [7]. Two localized algorithms were proposed. It was proved that the topologies generated by the proposed algorithms preserve bi-connectivity of networks. An extension of cone-based topology control algorithm was proposed in [1]. Each node decides its own power based on local information about relative angle of its neighbors. It showed that a fault-tolerant network topology is achievable and transmission power of each node is minimized to some extent. The proposed algorithm can be extended to 3-dimensions. All these works on topology control to construct fault-tolerant networks by adjusting transmit power of nodes. Movement of nodes is not a controllable parameter even in the works where mobile networks are considered.

Significant amount of work has been done in coordinating teams of mobile robots or actors. However, little attention was paid to incorporate fault tolerance into these robotic networks. For example, Dynia et al. [5] studied the problem of maintaining communication between an explorer robot and base station by moving other robots along the path.

Mobile robot network can be represented as a graph, where each node is a mobile robot and each edge denotes a communication link between a pair of robots. In a connected graph, a node is called a **critical node** if the graph is disconnected without the node. There are no critical nodes in a bi-connected graph. So, critical nodes are important in designing movement control algorithms to achieve bi-connected networks. Jorgic et al. [6] proposed an approach for localized p-hop critical node detection. To find if a node is critical in the network, a sub-graph of p-hop neighbors of the node is considered. From this sub-graph, the node itself and all its incident edges are excluded. If this resulting sub-graph of p-hop neighbors of a node is disconnected by excluding the node, then the node is critical. Since only local topological information is used, it is specified as p-hop critical node as it may not be globally critical. However, all the globally critical nodes are always p-hop critical for any value of p. As seen in Figure 1, the nodes A, B, and C are 2-hop critical nodes in the given network. We can also notice that nodes A and B in Figure 1 are only 2-hop critical nodes and are not globally critical. However, node C is globally critical in the network and is also 2-hop critical. Experiments showed that over 80% of locally estimated critical nodes and links are indeed globally critical [6].

Our problem is most related to the problem discussed by Basu and Redi [2], where movement control algorithms for fault-tolerant robot networks were proposed. In these algorithms, each mobile robot was assumed to be aware of global network topology. Based on the topological information, robots decide on their new position, which would thereby create a fault-tolerant network. The goal of these algorithms was to minimize the total distance travelled by all the robots. The authors further proposed an approximation algorithm for two dimensional cases. The basic idea was to divide a network into bi-connected blocks. The network is a block tree of these blocks. A block with maximum number of robots acts as the root of the tree. Algorithm works iteratively merging the blocks to form a single bi-connected block. Merging of the blocks is performed by block movement where each leaf block is moved towards its parent. If parent

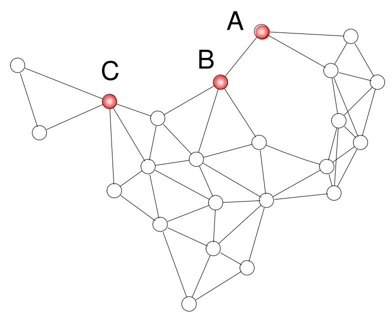


Fig. 1. An example of a network containing critical nodes.

block is empty then leaf block is moved towards a critical node. After each iteration, robot connectivity is recalculated and block tree is reconstructed as well. However, the proposed algorithms require accurate and global information of entire network. It is applicable to only small size networks. For large scale networks, not only is global network information hard to obtain and maintain, but also the total distance of movements and the communication overhead on robots increase rapidly.

3 Localized Movement Control

In this section, we propose a localized movement control algorithm for fault tolerance of mobile robot networks. To the best of our knowledge, it is the first localized movement control algorithm to achieve bi-connected network topologies. For simplicity, we use a node to denote a mobile robot for the rest of paper. We assume that all nodes in the network have a common communication range r. We further assume that each node has information of its p-hop neighbors. It is can be achieved by exchanging or relaying HELLO messages periodically within p-hops. To reduce exchange packets and collisions, we assume there is no RTS / CTS mechanism for transmissions of control packets. The network is assumed to be connected but not bi-connected. The problem of our concern is to control movement of nodes, such that the network becomes bi-connected. The objective is to minimize the total distance moved.

The distributed algorithm is executed at each node and starts as follows. At initialization stage, each node checks whether it is a p-hop critical node [6]. We define the **p-hop sub-graph** of a node by the graph which contains all nodes that are within p-hops from the node and all corresponding links. A node is said to be a **p-hop critical node** if and only if its p-hop sub-graph is disconnected without the node. Since each node is assumed to have knowledge of its p-hop sub-graph, it is able to determine whether it is a p-hop critical node. If a node finds itself a p-hop critical node, it broadcasts a *critical announcement* packet to all its direct neighbors.

To make the network bi-connected, all critical nodes should become non-critical by movement of nodes. Note that the movement of a node may create new neighbors, but it may also break some existing links. Since a critical node is the node that leaves its p-hop sub-graph disconnected without itself, breaking some current links of a critical node may cause disconnection of the network. However, for a non-critical node, the network remains connected if one of its current links is broken. Our basic idea of movement control is to move non-critical nodes while keep critical nodes static unless they become non-critical. According to number of critical neighbors of a critical node, there are three different cases that need to be considered. We start from the simplest case and discuss the three cases one by one.

3.1 Critical node without critical neighbors

In this case, a node finds itself a p-hop critical node and does not receive any *critical announcement* packet from its neighbors. Since it is a critical node, its p-hop sub-graph can be divided into two disjointed sets without the node. The basic idea is to select two neighbors from two sets respectively and move them towards each other until they become neighbors. Suppose distance between the two neighbors is d. Each node should move $(d - r)/2$ to reach each other. To minimize the total distance of movement of nodes, two neighbors with the minimum distance d among all possible pairs in the two sets are selected. The critical node sends these two neighbors a *movement control* packet containing their new locations. The two neighbors move to their new locations once the *movement control* packet is received. Note that a non-critical node may have several critical neighbors and it may receive multiple *movement control* packets from different critical nodes. Node IDs are used to assign priorities to critical nodes. Therefore, if a non-critical node receives more than one movement control packets, it always follows direction by the critical node having the largest ID. Note that there is no RTS/CTS mechanism in the network. The critical nodes with smaller IDs do not know and have no need to track the movement of their non-critical neighbors after sending *movement control* packets.

After movement of nodes, any node that loses a current neighbor or finds a new neighbor broadcasts a topology updated packet to its neighbors. This packet will be relayed hop by hop to reach p-hops neighbors of the sender. Each node receiving the topology updated packet updates its p-hop sub-graph and checks its new status. A new iteration of movement control begins. The movement control algorithm for case 3.1 is illustrated with the following example.

Consider the example shown in Figure 2, where node 3 in grey color is critical node and node 1, 2, 4, 5, 6, 7, 8 in white color are non-critical nodes. Suppose $p = 2$ in this example. Since node 3 is critical, its 2-hop sub-graph is divided into two disjointed sets $A = \{1, 2, 4, 5\}$ and $B = \{6, 7, 8\}$. Suppose distance of node 5 and 8 is the minimum among all possible pairs in these two sets, i.e. $d(5, 8) \leq d(x, y)$, $\forall x \in A, y \in B$. Node 3 computes new locations of node 5 and node 8 and sends *movement control* packets to them. Final locations of node 5 and node 8 are shown in Figure 2.

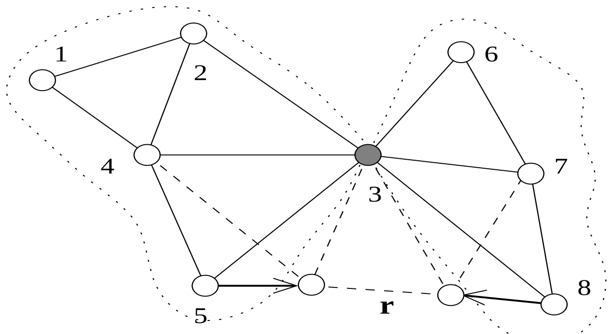


Fig. 2. Critical node without critical neighbor.

3.2 Critical node with one critical neighbor

In this case, there are two adjacent critical nodes and each critical node has only one critical neighbor. Suppose the two adjacent critical nodes are node 4 and node 5, and ID of node 5 is larger than ID of node 4. Our basic idea is to let the critical node with larger ID, node 5 in this case, select one of its non-critical neighbors to move towards the other critical node, node 4. Similar to case 3.1, node 5 divides its p-hop sub-graph into two disjointed sets. Node 4 is sure to be contained in one of the two sets. Node 5 searches the other set and selects one of its non-critical neighbors that is the nearest to node 4. Suppose distance between the selected neighbor and node 4 is d. The selected neighbor should move distance $d - r$ to reach node 4 since critical nodes are not allowed to move, to avoid disconnection of networks. Node 5 computes new location of its moving neighbor and sends it a *movement control* packet. The neighbor moves to its new location after receiving the *movement control* packet. Similar to case 3.1, node ID's are used to break the tie when a non-critical node receives multiple movement control packets. Once one of critical nodes becomes non-critical, we return to case 3.1 and movement control algorithm for case 3.1 is executed. Similar to case 3.1, topology update and checking status operations will start after movement of nodes. The algorithm for case 3.2 is illustrated with the following example.

Consider the example in Figure 3, where nodes 4 and 5 in grey color are critical nodes and nodes 1, 2, 3, 6, 7, 8 in white color are non-critical nodes. Since ID of node 5 is larger than ID of node 4, node 5 leads movement control. Suppose $p = 2$ again. Node 5 divides its 2-hop sub-graph into two disjoint sets $A = \{1, 2, 3, 4, 6\}$ and $B = \{7, 8\}$. Suppose distance of node 4 and 7 is the minimum among all neighbors in B. That is, $d(4, 7) \leq d(4, x)$, $\forall x \in B$. Node 5 computes new location of node 7, and sends it a *movement control* packet. Final location of node 7 is shown in figure 3.

Note that we cannot simply select one of node 5's neighbors that is the nearest to node 4 to move. It is because there may already exist links between node 4 and node 5's neighbors. For example in Figure 3, node 3 is the nearest node to

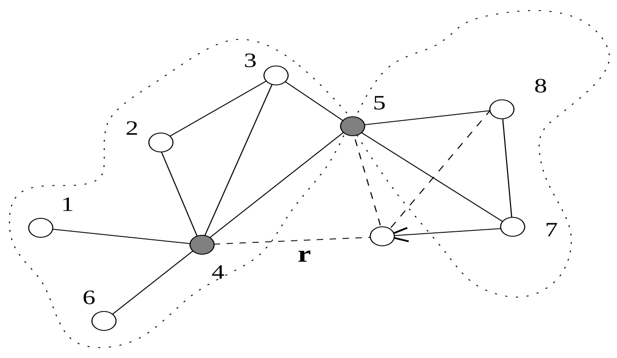


Fig. 3. critical node with one critical neighbor.

node 4 among all node 5's neighbors. However, they are already connected and there is no benefit of moving node 3 towards node 4.

3.3 Critical node with several critical neighbors

In this case, some critical node has more than one critical neighbor. Note that each node sends a *critical announcement* packet to all its direct neighbors if it finds itself a p-hop critical node. After that, all nodes in the network know the status of their neighbors. We say a critical node is **available** if it has non-critical neighbors and is **non-available** otherwise. A critical node is available means that it has non-critical neighbors that are able to move. An available/non-available critical node broadcasts an *available/non-available announcement* packet to its neighbors. A critical node declares itself a **critical head** if and only if it is available and its ID is larger than the ID of any available critical neighbor, or has no available critical neighbors. Our basic idea for general cases is to use the pair wise merging strategy. Simulation results show that this strategy can efficiently and quickly construct a bi-connected network. Each critical head selects one of its critical neighbors to pair with. Any criterion for selecting will work. To be deterministic, we decide that available critical neighbor (if any) with largest ID is selected, or otherwise non-available critical neighbor with the largest ID. Then the movement control algorithm for case 3.2 is called for each pair to compute the new topology.

Consider the example in figure 4, nodes 1, 2, 3, 4, 5, 6 in grey color are critical nodes (dashed block with a node is sub-graph of this node). Among these critical nodes, only nodes 1, 5, 6 are critical heads. Node 1 becomes a critical head since node 3 is non-available. Finally, there are three pairs: (1,3), (5,4) and (6,4), dominated by nodes 1, 5, and 6, respectively. Each critical head in a pair calls the movement control algorithm for case 3.2 to merge the pair. One can expect that the network density would increase after merging. Pair-wise merging continues until all critical nodes become non-critical, i.e., the network is bi-connected.

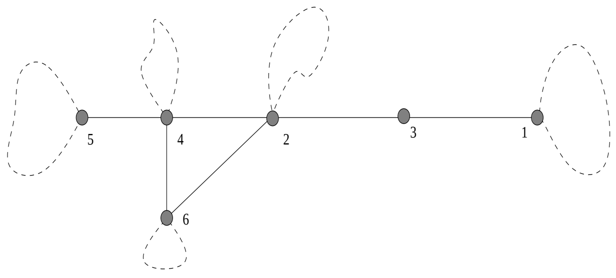


Fig. 4. Critical node with several critical neighbors.

Note that a critical head dominates a pair to merge at each time. No action will be taken if there is no critical head in the network. So the remaining problem is: does there always exist critical heads if the network is connected but not bi-connected? We answer this question below:

Lemma 1. *There exists non-critical nodes in any connected network.*

The Proof is trivial and has been omitted.

Theorem 1. *If the network is connected but not bi-connected then it has a critical node without critical neighbor or a critical head.*

Proof. According to Lemma 1, there exist non-critical and critical nodes in the network. Since the network is connected, some critical nodes will be then connected to some non-critical nodes. That is, some critical nodes are available. Consider among them the one with the largest ID. This node either has no critical neighbor or is a critical head by the definition.

According to Theorem 1, there always exist critical heads in the network if it is still not bi-connected. So, each node can iteratively call our algorithm until the network becomes bi-connected.

Note that we move only non-critical nodes in our algorithm. Controlled movement of a single non-critical node will never cause disconnection of networks. However, the network may be disconnected when multiple non-critical nodes move concurrently. It is because several non-critical nodes may form a *cut* of the network. The network becomes disconnected if critical links of nodes in the cut break. Unfortunately, the nodes of a cut can be distributed everywhere in the network. It is hardly possible for any localized algorithm to locally detect them. However, in our simulations, the proposed localized algorithm achieved 100% success on construction of bi-connected network topologies.

4 Performance Analysis

We tested the performance of our algorithm in a simulated environment and analyzed its efficiency with respect to the distance traveled metric. We also performed comparisons with the existing algorithm that uses global information [2] henceforth called the globalized algorithm. In all our simulations, the proposed algorithm was 100% successful, achieving bi-connectivity of the network within a few iterations, in most cases.

4.1 Simulation Environment

The main objective of our algorithm is to minimize the distance traveled by the robots in achieving bi-connectivity of the network. Thus, in our simulations we consider only the distance traveled metric and do not bother about the communication cost. The problem of minimizing the communication cost too, is part of the ongoing research conducted by our group. The results presented in this paper are based on simulations performed at the application layer, assuming an ideal MAC layer underneath, with no communication loss and instantaneous delivery of messages.

As for the simulation parameters, we considered sensor fields of sizes ranging from 20m X 20m to 100m X 100m, and the communication range of all the nodes were set to 10m. Nodes were placed randomly within the sensor fields, maintaining an average density of 1 node per $30m^2$ which ensures an average of around 10 neighbors per sensor node. We performed experiments varying the number of nodes in the sensor field while scaling the sensor field size accordingly. From the randomly generated networks, we selected the ones which were connected but not bi-connected and this set of networks were used in our experiments. We assume that each mobile robot has initial knowledge of its own position and can obtain information from its p-hop neighbors. We also assume that a robot can freely move from one position to another within the sensor field (i.e. there are no physical obstacles in the sensor field).

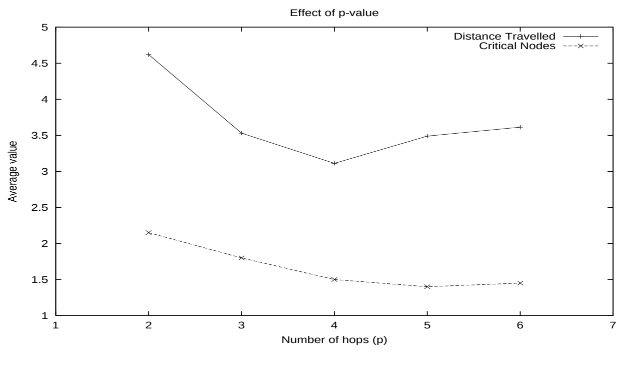
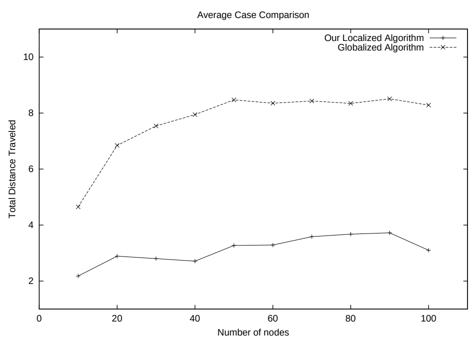


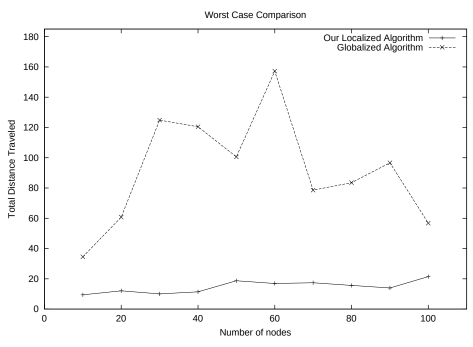
Fig. 5. Effect of the value of p on the distance traveled and the critical nodes identified by our algorithm.

4.2 Simulation Results

The p-value used in our algorithm is an important parameter that can be used to fine-tune the performance of our algorithm. Notice that if we use a small value of p, some nodes which are globally non-critical may be identified as critical by our localized algorithm. This will increase the value of distance traveled as more nodes need to be moved. We performed experiments with different values of p on a network of size 100, and the results are shown in Figure 5. As expected, both the total distance traveled and the critical nodes identified decrease with the increase in p-value. However, there is a sharp difference between $p = 2$ and $p > 2$, whereas the difference between $p = 3$ and $p > 3$ is not that significant. Since it is advisable to restrict the communication to smaller neighborhoods, we choose the value of $p = 3$, to balance the communication overhead and the distance traveled metric. The rest of the experiments were performed with the value of p set to 3.



(a)



(b)

Fig. 6. Comparison of the algorithms in networks of different sizes with fixed density and average degree 10. (a) Average case (b) Worst case

We performed extensive simulations to compare the performance of our algorithm with that of the globalized algorithm. The size of the network was varied from $n = 10$ to $n = 100$, and in each case, we repeated our experiments 100 times for both the algorithms. Figures 6 (a) and (b) show the average and worst case behavior of the two algorithms, in terms of the distance traveled metric. Our proposed algorithm outperforms the globalized algorithm significantly in all cases. In our algorithm, in each iteration, individual nodes are moved towards each other instead of moving blocks of nodes together as in the case of the globalized algorithm. This results in great performance improvement as can be seen from the simulation results. Notice that, since our algorithm uses only local information to identify the critical nodes, it would identify more nodes as critical nodes as compared to the globalized algorithm. Figure 7 compares the number of nodes initially identified as critical by the two algorithms, in the average case.

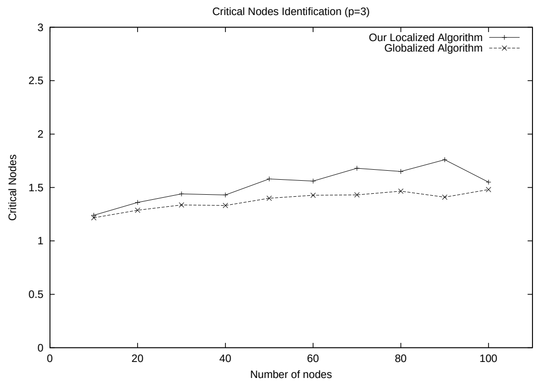


Fig. 7. Critical node identification by the two algorithms in networks of various sizes

It seems that our localized algorithm did not identify too many globally non-critical nodes as critical, which suggests that using only local information is not too harmful in general, for identifying critical nodes.

5 Conclusions and Future Work

In this paper, we proposed a localized movement control algorithm to construct a fault-tolerant mobile robot network. We presented simulations results to show the effectiveness of our algorithm and its efficiency in terms of the total distance travelled by the robots. The simulations results for randomly generated networks show that our localized movement control algorithm significantly outperforms its globalized counterpart. It is interesting to note that, in most cases, the use of local information (in fact, information about 3-hop neighbors only) is sufficient to convert the network to a bi-connected one in an efficient manner. Thus, global information about the network is not necessary to achieve bi-connectivity. The results shown in this paper are for networks obtained by randomly scattering robots on fixed region and then selecting those which satisfy the condition of connectivity and non-bi-connectivity. On this class of graphs, the proposed algorithm seems to work quite well, in fact achieving 100% success. However, it is quite possible that for some specific class of networks this algorithm may not perform that well. In particular, we tested only networks with average number of neighbors (density) about 10, and results may differ for lower densities. In future, we plan to do more extensive simulations on specially created network topologies in order to identify possible classes of networks for which our algorithm may give poor performance. We are also investigating on applications of mobile robots in sensor networks as data collectors. We recognize that mobile robots can also be used in heterogeneous sensor networks as data collectors. In these applications, sensor field coverage is an important metric for the algorithm and would be the topic for future work.

The localized algorithm proposed in this paper achieves fault-tolerance by converting a connected network to bi-connected one. In case the original network is disconnected, the algorithm can be used to make each connected component fault-tolerant. However, the problem of constructing a connected and fault-tolerant network starting from a disconnected network is much more difficult and would be considered in future. Finally, we did not consider the problem of minimizing communication cost as added criterion, although it was addressed by localized design. That problem is currently under investigation by this group.

Acknowledgements

This research is supported by NSERC Strategic Grant STPGP 336406, and the UK Royal Society Wolfson Research Merit Award.

References

1. M. Bahramgiri, M. Hajiaghayi, and V. Mirrokni. "Fault-tolerant and 3-dimensional distributed topology control algorithms in wireless multi-hop networks", In Proc. IEEE Int. Conf. on Computer Communications and Networks (ICCCN02), pp. 392–397, 2002.
2. P. Basu and J. Redi, "Movement control algorithms for realization of fault-tolerant ad hoc robot networks" , IEEE Network, 18(4):36-44, 2004.
3. R. Chow, and T. Johnson, "Distributed Operating Systems & Algorithms" , Addison Wesley Longman Inc., 1997.
4. J. Cheriyan, S. Vempala, and A. Vetta, "An Approximation Algorithm for the Minimum-Cost k-Vertex Connected Subgraph" , SIAM J. Computing 32(4): 1050-1055, 2003.
5. M. Dynia, J. Kutylowski, P. Lorek, and F.M. auf der Heide, "Maintaining Communication Between an Explorer and a Base Station" , in IFIP Conf. on Biologically Inspired Cooperative Computing (BICC'06), vol. 216, pp. 137-146, 2006.
6. M. Jorgic, I. Stojmenovic, M. Hauspie, D. Simplot-Ryl, "Localized Algorithms for Detection of Critical Nodes and Links for Connectivity in Ad Hoc Networks " , In 3rd Ann. Med. Ad-Hoc Workshop (Med-Hoc-Net), pp. 360-371, 2004.
7. N. Li, and J.C. Hou, "Topology Control in Heterogeneous Wireless Networks: Problems and Solutions" , In Proc. IEEE INFOCOM, pp. 232-243, 2004.
8. N. Li, and J.C. Hou, "FLSS: A Fault-Tolerant Topology Control Algorithm for Wireless Networks", in Proc. 10th Ann. Int. Conf. on Mobile Computing and Networking, pp. 275-286, 2004.
9. H. Liu, P.J. Wan, X. Jia, "Fault-Tolerant Relay Node Placement in Wireless Sensor Networks" , in Proc. COCOON, pp. 230-239, 2005.
10. R. Ramanathan and R. Rosales-Hain, "Topology Control of Multihop Wireless Networks using Transmit Power Adjustment", In Proc. IEEE Infocom, vol.2, pp. 404-413, 2000.
11. D. Vass, A. Vidcs, "Positioning Mobile Base Station to Prolong Wireless Sensor Network Lifetime" , in CoNEXT 2005 Student Workshop, pp. 300-301, 2005.
12. O. Younis, S. Fahmy, and P. Santi, "Robust Communications for Sensor Networks in Hostile Environments", In Proc. 12th IEEE Int. Workshop on Quality of Service (IWQoS), pp. 10-19, 2004.