Volume 3, Issue 11, November 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Fault Tolerance in Wireless Adhoc network through Biconnectivity

Manjula Srinivas*, B.Renuka, N.Jayalakshmi
CSE & JNTUK
India

Abstract*—A wireless ad hoc network consists of a collection of transceivers, in which a packet may have to traverse multiple consecutive wireless links to reach its final destination. Wireless networks face a variety of constraints that do not appear in wired networks. There are also fault tolerance requirements, due to their evolving critical application domains and to the large number of failures that may result from mobility, fading, or obstructions. A biconnected communication graph is necessary to ensure fault tolerance in wireless network, since wireless ad hoc networks are used in critical application domains where failures are likely to occur. This means that each pair of nodes in the network has at least two node disjoint paths between them, and thus, failure at any single node does not partition the network. In this paper, we first introduce a new concept of removable node, such that the removal of it does not partition the network anymore. Algorithms are provided to identify removable nodes, in which all nodes can be simultaneously removed from the network without generating any new cut-node in the network, and to derive the final location of a removable node such that its movement distance is the shortest and the associated cut-node becomes a non-cut-node.

***Keywords*— Wireless ad hoc networks, biconnectivity, fault tolerance, cut-node.**

I. INTRODUCTION

An *ad hoc network* consists of a collection of transceivers, in which a packet may have to traverse multiple consecutive wireless links to reach its final destination. Wireless networks face a variety of constraints that do not appear in wired networks. There are also fault-tolerance requirements, due to their evolving critical application domains and to the large number of failures that may result from mobility, fading, or obstructions [1]. Wireless ad hoc networks are also popularly applied as sensor networks and robot networks [2]–[6]. In all these applications, each node has the functionality of both a terminal (to generate/receive traffic to/from the network) and a relay (to help forward traffic of other nodes to their destinations). In this context, if a node is unable to function well (e.g., due to energy depletion or due to malicious attacks), which is referred to as a *fault*, communication failure or even network partition (i.e., the whole network is partitioned into two or more isolated sub-networks) may happen. A connected graph is usually assumed as the minimum connectivity requirement by the algorithms running in different layers of the network, such as routing protocols [7]. However, if there is only one path between a pair of nodes, failure of a single node (or link) between them will result in a disconnected graph. Therefore, topologies with multiple, alternative disjoint paths between any pair of nodes are often required [8]. It has been well accepted that bi-connectivity is a desired fault-tolerance feature. The *transmission graph* G = (V,E), where E ={(u, v) : u $\in$ V, v $\in$ V} is said to be biconnected if, for any two nodes u, v V , there exist two node-disjoint paths connecting u to v. Since a biconnected graph is also bi edge connected, but the converse is not necessarily true, we say that a graph is biconnected if it is 2- node connected. This problem was proved to be NP-hard by Calinescu and Wan [9]. In a bi-connected network, if a fault happens at any node, the network is still connected, i.e., all affected communication links can be re-routed. In other words, the removal of a node and associated links from a bi-connected network does not partition the network. On the other hand, if a network is partitioned after the removal of a node and associated links, then the node is called a *cut-node*. Nodes that are not cut-nods are called *non-cut-nodes*. Therefore, to achieve biconnectivity, actions should be taken to make all cut-nodes in the network become non cut-nodes. We first introduce a concept of *removable node* (the removal of which will neither partition the network nor generate any new cut-node in the network). If we let a removable node move to the position of a cut-node, a fault at the cut-node will not partition the network anymore, and thus, the cut-node will become a non cut-node. Based on this idea, we present an algorithm to identify removable nodes, an algorithm to match the cut-nodes with a feasible set of removable nodes (simultaneously removing of which will not generate new cut-node), and an algorithm to decide each removable node's final location (which is not necessarily the matched cut-node's position) with the shortest movement distance .

II. ACHIEVING BICONNECTIVITY

Biconnectivity is a desirable property for a network to have for fault tolerance. Wireless ad hoc networks are popularly applied as sensor networks and robot networks [2]–[6]. In all these applications, each node has the functionality of both a terminal (to generate/receive traffic to/from the network) and a relay (to help forward traffic of other nodes to their destinations). In this context, if a node is unable to function well (e.g., due to energy depletion or due to malicious attacks), which is referred to as a *fault*, communication failure or even network partition (i.e., the

© 2013, IJARCSSE All Rights Reserved

whole network is partitioned into two or more isolated sub-networks) may happen. This necessitates movement for some of them in order to *create* extra links such that the resultant topology is biconnected. Figure 1 illustrates the idea.

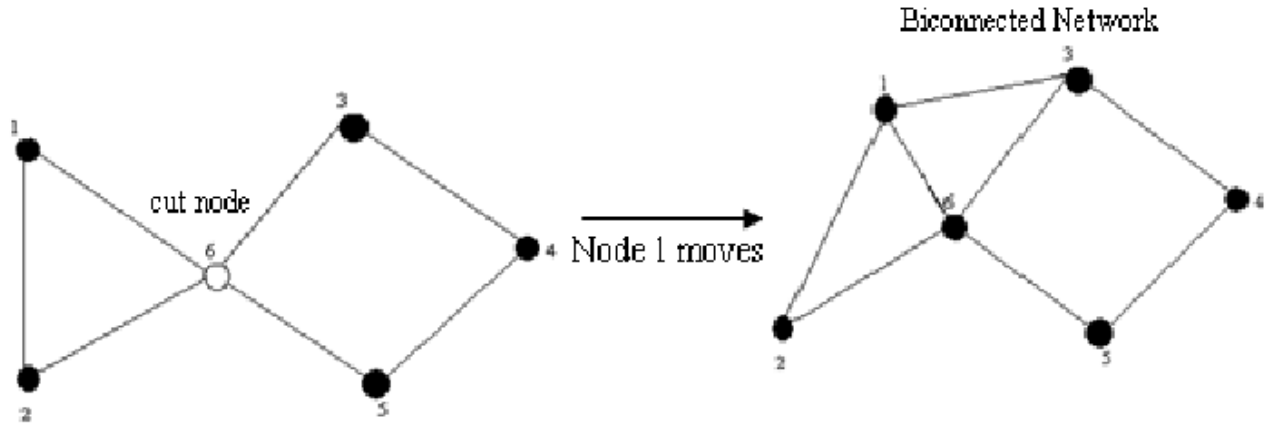


Figure 1: Achieving Biconnectivity by Node Movement

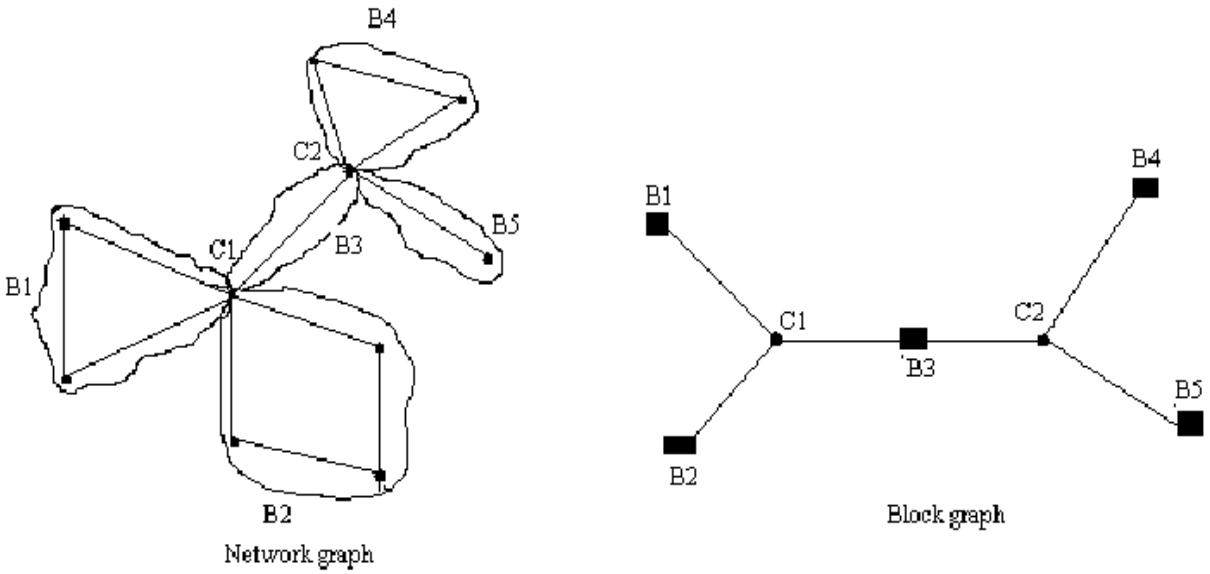
For a network graph, a non-cut-node is called a *removable node* if removal of the node does not generate any new cut-node in the network graph. Based on the example in Fig. 1, to make a cutnode in the network become a non-cut-node, we can move a removable node (such as node 6 Fig. 1) to the position of the cut-node. No new cut-node will be generated after the movement.

III. IDENTIFICATION OF REMOVABLE NODE

Consider a wireless ad hoc network with a number of nodes. The communication range of each node is *R*. So if the distance between two nodes is less than *R*, the two nodes can directly communicate to each other. The network can be represented by an undirected graph G = {V, E}, where V is the set of vertices (each vertex representing a node) and E is the set of edges. For any two vertices in V, denoted *a* and *b*, if the distance of their represented nodes is less than *R* (i.e., the two nodes can directly communicate), there is an edge in E connecting the two vertices, denoted *e*(*a*, *b*).

If graph G is connected, a block either is a bridge with its ends (when the number of nodes in the block is two), or is bi-connected (when the number of nodes in the block is more than two). Any two blocks of G have at most one common node, which, if exists, is a cut-node of G. This means in G, each edge belongs to a unique block, and each non-cut-node also belongs to a unique block [10].

• *Block-graph*: Let *B* denote the set of blocks in the network, and **C** denote the set of cut nodes in the network. Note that each cut-node belongs to two or more blocks. Then the *block graph* [10] of the network can be formed on **C**/*B* such that if a cut-node belongs to a block, then there is an edge in the block graph that connects the cut-node and the block. For a connected network, the block graph is a bipartite graph, which is also a tree [10].



An example connected network and its block graph are shown in Fig. 2, which has five blocks (B1, B2, B3, B4, B5) and two cut-nodes (*c*1 and *c*2). A leaf vertex in the block graph should be a block. It is important to identify removable nodes in a network. Algorithms are given in [11] to identify cut-nodes. Although the definition can be used to identify removable nodes, global network topology information is needed, which is not desired. In the following, methods are given to demonstrate when node *i* can be identified as a *removable node* in the network.

© 2013, IJARCSSE All Rights Reserved

Condition for a Removable Node:

A non-cut-node belongs to one and only one block of the network [10]. A removable node should be at first a non-cut-node. Let G = {V, E} denote the original network graph, and $G^{\backslash i} = \{V^{\backslash i}, E^{\backslash i}\}$ denote the network graph after removal of node *i*. Let B*i* denote the block of G that includes non-cut-node *i*, and $Bi^{\backslash i}$ denote the remaining part of B*i* after node *i* is removed from B*i*.

Theorem 1: A sufficient and necessary condition for a non-cut-node *i* being a removable node in G is:
Any cut-node in $Bi^{\backslash i}$ is also a cut-node in G

*Proof:*We first prove the sufficiency of the condition. We use proof by contradiction.
Assume any cut-node in $Bi^{\backslash i}$ is also a cut-node in G, but node *i* is not a removable node in G. Then a new cut-node, denoted node *l*, is generated in G*i*, which is a non-cut-node in G. According to Lemma 1(If the removal of a non-cut-node *i* from G generates new cut-nodes in G*i*, then all the new cut-nodes in G*i* belong to $Bi^{\backslash i,}$ and are also cut-nodes in $Bi^{\backslash i}$), node *l* belongs to $Bi^{\backslash i}$ and is a cut-node in $Bi^{\backslash i}$. According to our assumption, node *l* is also a cut-node in G. This contradicts the fact that node *l* is a non-cut-node in G. Therefore, the sufficiency of the condition in this theorem is proved.
From the tree structure of the block graph of G, it can be seen that: if removal of a node from $Bi^{\backslash i}$ partitions $Bi^{\backslash i}$, removal of the node also partitions G*i*. This means a cut-node in $Bi^{\backslash i}$ is also a cutnodein $G^{\backslash i.}$ On the other hand, since node *i* is a removable node in G, any cut-node in $G^{\backslash i}$ is a cut-node in G. Based on these two observations, it can be seen that any cut-node in $Bi^{\backslash i}$ is a cutnode in G.

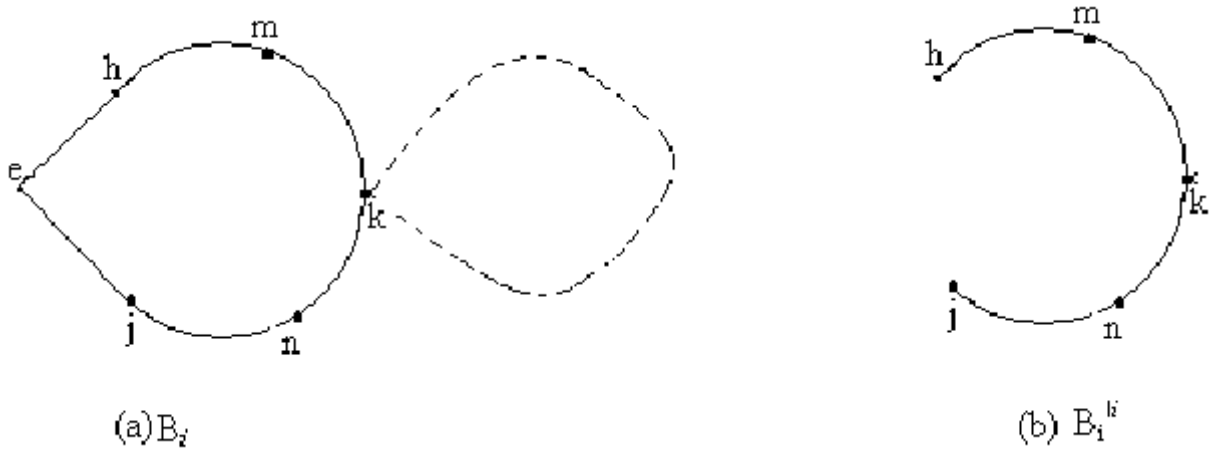


Fig. 3. Illustration examples of Theorem 1.

We use Fig. 3 to illustrate Theorem 1. Fig. 3(a) shows block B*i*, which is a ring. The dashed line means other part of the network G. It can be seen that nodes *h, I, j,m* and *n* are non-cut nodes, while node *k* is a cut-node, of the network. Fig. 3(b) presents subgraph $Bi^{\backslash i}$ where node *I* is removed. We can see nodes *m* and *n*, which are non-cut-node in G, are cut-nodes of subgraph $Bi^{\backslash i}$. From Theorem 1, node *i* is not a removable node of G. To identify whether a target non-cut-node is a removable node, we only need to focus on the block that includes the node rather than on the whole network.

Algorithm to identify removable node

Recall that, when Theorem 1 is used to identify removable nodes, we can use only local information (i.e., *p*-hop local topology information) at the tested nodes, which is desired for a algorithm1. Although not all the removable nodes can be identified based on Theorem 1, a significantly large portion of the removable nodes can be identified. In this algorithm, the Depth-First-Search (DFS) algorithm [12] is used to find blocks of the *p*-hop local topology. The larger value of *p*, the larger local topology of the target node, the larger blocks that can be found, and the more removable nodes that can be identified. In a wireless ad hoc network, there may exist multiple cut-nodes. So we may need to move multiple removable nodes to deal with the cut-nodes. However, if two nodes both are removable nodes, it is still likely that removal of one node may make the other not a removable node anymore. The algorithm1 can be used to identify *individual* removable nodes.

Algorithm1: Determination of node *i* as a removable node.
1: if the neighbor degree of node *i* is one then
2: node *i* is a removable node.
3: return.
4: if the neighbor degree of node *i* is two then
5: if the two neighbors of node *i* are neighbor to each other then
6: node *i* is a removable node.

© 2013, IJARCSSE All Rights Reserved

7: return.
8: create the local topology of node *i* based on its *p*-hop neighbors.
9: execute the Depth-First-Search (DFS) algorithm [13] to the local topology.
10: if the local topology is connected then
11: get all blocks of the local topology. Denote the number of blocks as *m*.
12: for *k* = 1: *m* do
13: if block *k* includes all of node *i*'s neighbors then
14: node *i* is a removable node.
15: return.

Removable Nodes Moving Strategy

We can use Algorithm 2 to find a nearby removable node in a distributed way, which has two
Phases: initialization phase and iteration phase.

Initialization phase:

Each node obtains information of its *p*-hop neighbors by exchanging HELLO messages within *p*-hops [14]. Each node in the network executes the distributed block finding algorithm in [5 to identify whether it is a cut-node. Particularly, each block has a label for identification. The blocking finding algorithm associates each edge with the label of the block that the edge belongs to. Therefore, if a node's all edges are associated with a same block label, then the node is a non cut-node; and if the node's edges are associated with multiple block labels, then the node is a cut-node.

Iteration Phase: The iteration phase consists of a number of iterations. Each iteration is as follows:-

Each non-cut-node executes Algorithm 1, based on its *p*-hop local topology, to determine whether it is a removable node. Each cut-node broadcasts a request (REQ) message to its neighbors. Upon reception of an REQ message, a non-cut-node forwards the message to its neighbors. Note that a cut-node does not help relay any REQ message. Therefore, for each cut node, its REQ message is broadcast only to blocks (of the network) that include the cut-node. Each removable node, upon reception of all REQ messages in its block, selects one cut-node with the minimum ID, and sends a response (REP) message to the cut-node. Each cut-node, upon reception of all REP messages from its blocks, selects one associated removable node that has the shortest distance to itself, and sends a notification (NTF) message to the removable node. Then, the selected removable node is matched with the cut-node, and moves to the location of the cut-node or to a location with the shortest movement distance as discussed in Algorithm 3.

Since the cut-node becomes a non-cut-node, its associated blocks should be merged into one block. Therefore, after reaching it final location, the originally removable node sends a MERGE message to its new neighbors and upon reception of the MERGE message, each non cut- node helps forward the message to its neighbors. Each non-cut-node, upon reception of the MERGE message, associates its edges with the new block label that is included in the MERGE message. Note that before and after the movement of the removable node, it sends a TOPOLOGY UPDATE message to its old neighbors and new neighbors, respectively. The TOPOLOGY UPDATE message is forwarded within *p*-hop neighbors of the sender. Any node, upon reception of the message, updates its *p*-hop local topology. The above procedure is repeated in all subsequent iterations, until all cut-nodes become non cut-nodes, or until no response to REQ messages. If the latter situation6 happens, it means the final bi-connectivity is not reached. In this case, other algorithms may be resorted, such as the block movement algorithm [24], to deal with the remaining cut-nodes.

Distance movement by nodes

For a removable node to deal with a cut-node, it may move to the position of the cut-node. However, actually it is not necessary for the removable node to move to the exact location of the cut-node.

Algorithm 3: Shortest Movement Distance Algorithm of Removable node *i* to Deal with Cut- Node *l*.
1: through the DFS algorithm, find all the connected components of node *l*'s local topology.
2: take one node from each connected component, and form a set. There are totally *K* such sets, denoted *c*1, *c*2,…., *ck*.
3: let the shortest movement distance of node *i* be *d*opt = ∞ and denote the optimal new location of node *i* as (*x*opt , *y*opt).
4: for *k* = 1 : *K* do
5: obtain the minimal enclosing circle for all nodes in set $c_{k.}$
6: if the radius of the circle < *R* then
7: solve the optimization problem: Minimize(*x*,*y*) $\sqrt{(x-xi)^2+(y-yi)^2}$

subject to: $\sqrt{(x-xj)^2+(y-yj)^2}$ < *R;* $\forall j \in c_{k.}$ in which (*xi*, *yi*) means the original location of node *i*.
8: in the solution of the optimization problem, denote the new location of node *i* as

($x^{(k)}$ opt, $y^{(k)}$ opt), and $d^{(k)}$opt = $\sqrt{(x(k)\ \text{opt}-xi)^2+(y(k)\ \text{opt}-yi)^2}$

9: if $d^{(k)}$opt < *d*opt then
10: $d^{(k)}$opt-> *d*opt, ($x^{(k)}$opt, $y^{(k)}$ opt) -> (*x*opt, *y*opt).
11: (x_{opt}, , y^{opt}) is the new location of the removable node *i* with the shortest movement distance.

© 2013, IJARCSSE All Rights Reserved

The Fig. 4 shows the execution of algorithm3.

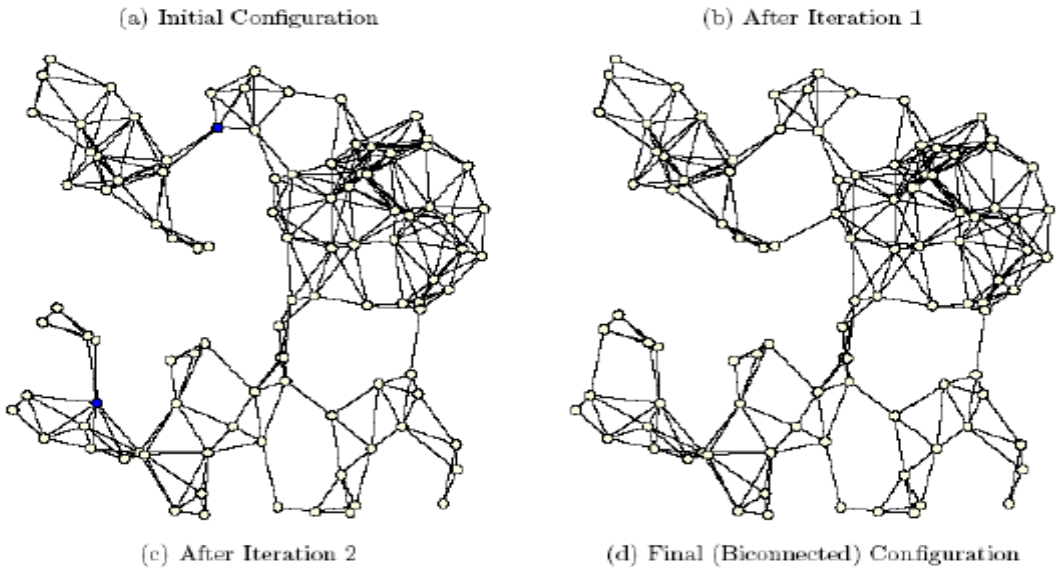


Figure 4: Execution of the Block Movement Algorithm

IV. CONCLUSION

Wireless Ad hoc networks have become an increasingly important object of study, due to their many applications in situations where wired backbones are infeasible or economically inefficient. Fault tolerance is an extremely desirable property in network design, and biconnectivity is a baseline feature in that domain. Fault tolerance can be achieved by means of moving nodes to locations which results in richer topologies. At the same time nodes should move as little distance as possible in so far the desired topological property is achieved. In this paper, we proposed simple algorithms for moving nodes to new locations such that the resulting network becomes biconnected. Make Biconnected on the other hand attempts to remove cut vertices from the graph in a systematic iterative manner. In this paper, we provided helpful insights for fault tolerance design in wireless ad hoc networks.

REFERENCES

[1]. A. Srinivas and E. Modiano, "Minimum energy disjoint path routing in wireless ad-hoc networks," in *Proceedings of the 9th Annual International Conference on Mobile Computing*

[2]. Haas ZJ, Deng J. Dual busy tone multiple access (DBTMA)³a multiple access control scheme for ad hoc networks. *IEEE Transactions on Communications* 2002; 50(6): 975–985.

[3]. Jiang H,Wang P, Poor HV, Zhuang W. A distributed MAC scheme supporting voice services in mobile ad hoc networks. *Wireless Communications and Mobile Computing* 2010; 10(4): 547–558.

[4]. Wasef A, Lu R, Lin X, Shen X. Complementing public key infrastructure to secure vehicular ad hoc networks. *IEEE Wireless Communications* 2010; 17(5): 22–28.

[5]. Wang P, Jiang H, Zhuang W. A new MAC scheme supporting voice/data traffic inwireless ad hoc networks. *IEEETransactions on Mobile Computing* 2008; 7(12): 1491–1503.

[6]. Zheng J, Wang P, Li C. Distributed data aggregation using Slepian-Wolf coding in cluster-based wireless sensor networks. *IEEE Transactions on Vehicular Technology* 2010; 59(5): 2564–2574. *Networking*, San Diego, 2003, pp. 122–133.

[7]. R. Madan and S. Lall, "Distributed algorithms for maximum lifetime routing in wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 5, pp. 2185–2193,2006.

[8]. M.K . Marina and S. R. Das, "On-demand multipath distance vector routing in ad hoc networks," in *Proceedings of the 9th International Conference on Network Protocols*, Riverside,2001, pp. 14–23.

[9]. G. Calinescu and P. J. Wan, "Range assignment for biconnectivity and k-edge connectivity in wireless ad hoc networks," *Mobile Network and Applications*, vol. 11, pp. 121–128, 2006.

[10]. Diestel R. *Graph Theory*, 3rd Ed. Springer-Verlag, Heidelberg, 2006.

[11]. Jorgic M, Stojmenovic I, Hauspie M, Simplot-Ryl D. Localized algorithms for detection of critical nodes and links for connectivity in ad hoc networks. In *Proceedings f 3rd Annual Mediterranean Ad Hoc Networking Workshop*, Bodrum, Turkey, June 2004; 360–371.

[12]. Cormen TH, Leiserson CE, Rivest RL, Stein C. *Introduction to Algorithms, 2nd Ed.* MIT Press, 2001.

[13]. Das S, Liu H, Nayak A, Stojmenovic I. A localized algorithm for bi-connectivity of connected mobile robots. *Telecommunication Systems* 2009; 40(3-4): 129–140.

[14]. Hohberg W. How to find biconnected components in distrubited networks.*Journal of Parallel and Distributed Computing*

© 2013, IJARCSSE All Rights Reserved