

# Disjoint Multipath Routing in Dual Homing Networks using Colored Trees

Preetha Thulasiraman, Srinivasan Ramasubramanian, and Marwan Krunz  
Department of Electrical and Computer Engineering  
University of Arizona, Tucson, AZ 85721  
{pthulasi, srini, krunz}@ece.arizona.edu

**Abstract**— Wireless sensor networks (WSNs) employed in monitoring applications require data collected by the sensors to be deposited at specific nodes, referred to as *drains*. To improve robustness in data collection, we consider a dual homing network in which two drains are employed and every node is required to send data to the two drains over link- or node-disjoint paths. One approach to reduce the number of routing table entries at a node is to construct two trees, namely red and blue, each rooted at a particular drain such that the paths from any node to the two drains on the trees are link- or node-disjoint. In this paper, we develop the first distributed algorithm for constructing colored trees in a dual-homing network whose running time is linear in the number of links. In addition, we show that the average path length may be optimized by employing the generalized low-point concept rather than the traditional low-point concept.

## I. INTRODUCTION

Recent advancements in low-power computing, sensing, and wireless communications have contributed to the emergence of multi-hop wireless sensor networks (WSNs) as a cost-effective solution for surveillance/monitoring applications. The data collected by sensors in a WSN is required to be deposited at specific nodes, referred to as *drains*. In order to improve robustness in data collection, we consider a scenario where two drains are employed and data from every sensor is required to be deposited at both the drains. As the sensor nodes have limited battery energy and computational capabilities, implementation of sophisticated transport layer protocols that guarantee end-to-end reliable transmission is impractical. One approach to achieve the goal of robustness is to employ multipath routing (MPR). MPR operates by transmitting data over multiple paths. In general, the multiple paths from a source to a destination may have common links (or nodes) as long as the shared links (or nodes) have sufficient resources. To improve the transmission reliability and avoid shared-link (or node) failures, the multiple paths can be selected to be link- or node-disjoint. In this case, the MPR approach is referred to as *disjoint multipath routing* (DMPR).

Implementation of MPR and DMPR poses two main challenges [1]. The first is related to the computation of loop-free multiple paths. For large networks, a distributed solution that relies only on local information is preferred. Distributed multipath routing algorithms in the literature are developed in the context of wireless networks. MPR approaches based on Dynamic Source Routing (DSR) [2], [3], [4] require the destination to select maximally disjoint paths among the

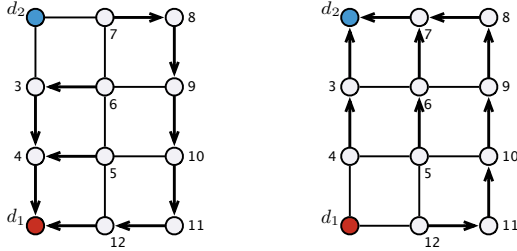
received route requests. MPR approaches based on AODV routing [5], [6], [7], [8], [9] do not guarantee finding disjoint paths. Protocols such as Directed Diffusion [10] that have been developed specifically for wireless sensor networks do not guarantee disjoint multiple paths and may result in loops after a link/node failure.

The second challenge of implementing MPR or DMPR techniques is related to forwarding of data over the multiple paths. Datagram networks rely on the destination address in the packet header for forwarding packets over one path. To implement MPR or DMPR techniques, every node must maintain a set of preferred neighbors to reach a destination, such that the paths are loop-free (and disjoint, if needed). The forwarding of packets must be based on destination address and some “additional” information (such as source address, labels, etc.). The intermediate nodes must be aware of this additional information or otherwise, it must be carried in every packet header.

To reduce the routing table overhead, hence reduce lookup time, a novel multipath routing strategy called *colored trees* (CT) was developed [11]. Every node in the network has two preferred neighbors to a destination, namely red and blue. A packet transmitted from a source is marked with one of the two colors. An intermediate node that receives the packet forwards it to its preferred neighbor based on the color of the packet. Thus, the routing table at a node has only two entries for every destination node. The network may be viewed as two trees, namely red and blue, that are rooted at the destination. The two paths from a given source to the root of the two trees are link/node-disjoint. The colored trees may be constructed by adapting the centralized algorithms developed for robust multicasting [12], [13], [14], by simply reversing the arcs obtained in the multicast trees. The first distributed algorithm for constructing colored trees, whose complexity is linear in the number of edges, was developed in [1].

A dual homing network (DHN) is traditionally employed in IP-based access networks [15]. A DHN consists of two dedicated routers, referred to as *dual homes*, through which the nodes within the access network are connected to those outside the access network. A sensor network with two drains may be viewed as a DHN where the two drains act as the two homes. In order to overcome a single link (or node) failure within the network, every node is required to have a path to the two homes that are link-disjoint (or node-disjoint). In order to

maintain disjoint routes with minimum routing table overhead, thereby reducing routing table lookup time, we develop the colored trees to dual homes (CTDH). In this approach, the red tree is rooted at one home while the blue tree is rooted at the other such that the path from any source to the two homes on the two trees are link-disjoint (or node-disjoint). Figure 1 shows two trees, one rooted at node  $d_1$  and the other rooted at  $d_2$ , in a dual-homing network with nodes  $d_1$  and  $d_2$  as the dual homes. It may be observed that the paths from any node to the dual homes on the two trees are node-disjoint.



(a) Red tree rooted at  $d_1$ . (b) Blue tree rooted at  $d_2$ .

Fig. 1. An example network with red and blue trees. The path from a source to the two homes (drains) are node-disjoint on the two trees.

**Contribution.** This paper (1) proves that 2-edge (or 2-node) connectivity is a sufficient condition for the existence of a solution to the CTDH problem satisfying link-disjoint (or node-disjoint) constraint; (2) develops a distributed solution to the CTDH problem whose complexity is linear in the number of edges; (3) evaluates the effectiveness of employing generalized low-point concept compared to traditional low-point concept; and (4) evaluates the effectiveness of the trees constructed under multiple link failures.

The rest of the paper is organized as follows: Section II describes the network model and problem definition. Section III develops the linear-time distributed algorithm for constructing the two colored trees. Section IV presents the performance comparison of the distributed algorithm with traditional and generalized low-point concepts. Our conclusions are presented in Section V.

## II. PROBLEM STATEMENT AND PRELIMINARIES

Consider a network  $\mathcal{G}(\mathcal{N}, \mathcal{L})$  composed of a set of nodes  $\mathcal{N}$  and a set of links  $\mathcal{L}$ . The links are assumed to be bi-directional. The terminology of *arc* is used to refer to a directed link between two nodes. An arc from node  $i$  to  $j$  is represented as  $i \rightarrow j$ . Given dual homes  $d_1, d_2 \in \mathcal{N}$ , the goal is to construct two trees  $\mathcal{R}$  and  $\mathcal{B}$  (referred to as the red and blue trees, respectively) rooted at  $d_1$  and  $d_2$ , respectively, that minimize the average path length from a source to the homes such that the CTDH-LD (CTDH-ND) version of the problem satisfies the link-disjoint (node-disjoint) path constraint. The disjoint path constraints are stated as follows. Let  $\mathcal{P}_{sd_1}^{\mathcal{R}}$  and  $\mathcal{P}_{sd_2}^{\mathcal{B}}$  denote the paths from a node  $s$  to drains  $d_1$  and  $d_2$  on trees  $\mathcal{R}$  and  $\mathcal{B}$ , respectively.

**Link-disjoint path constraint:**

$$\forall s \in \mathcal{N} \setminus \{d_1, d_2\} \text{ and } \forall i, j \in \mathcal{N}$$

$$i \rightarrow j \in \mathcal{P}_{sd_1}^{\mathcal{R}} \Rightarrow (i \rightarrow j \notin \mathcal{P}_{sd_2}^{\mathcal{B}}) \wedge (j \rightarrow i \notin \mathcal{P}_{sd_2}^{\mathcal{B}}).$$

**Node-disjoint path constraint:**

$$\forall s \in \mathcal{N} \setminus \{d_1, d_2\} \text{ and } \forall i \in \mathcal{N} \setminus \{s, d_1\}$$

$$i \in \mathcal{P}_{sd_1}^{\mathcal{R}} \Rightarrow (i \notin \mathcal{P}_{sd_2}^{\mathcal{B}}).$$

If the two homes are the same, say  $d_1 = d_2 = d$ , then we refer to the problem of constructing colored trees rooted at a given destination  $d$  such that the path from any node to the destination  $d$  is link-disjoint (node-disjoint) as the CT-LD (CT-ND) problem.

**Theorem:** A solution to the CTDH-LD (CTDH-ND) problem exists if the network is 2-edge (2-node) connected.

**Proof:** Construct a graph  $\mathcal{G}'$  from  $\mathcal{G}$  with an additional node  $d$  and two bidirectional links  $(d, d_1)$  and  $(d, d_2)$ . The graph  $\mathcal{G}'$  remains 2-edge (2-node) connected if the graph  $\mathcal{G}$  is 2-edge (2-node) connected. Since  $\mathcal{G}'$  is 2-edge (2-node) connected, a solution to the CT-LD (CT-ND) problem exists [12] with node  $d$  as the destination. As node  $d$  has only two links, all the paths on one tree must traverse node  $d_1$  and link  $d_1 \rightarrow d$  as the last node and link, respectively. Similarly, all the paths on the other tree must traverse node  $d_2$  and link  $d_2 \rightarrow d$ . Removing node  $d$  and the links attached to it results in a solution to the CTDH-LD (CTDH-ND) problem, thus proves the theorem.

The CTDH-LD (CTDH-ND) problems may be formulated as an integer linear program (ILP) by extending the formulation of the CT-LD (CT-ND) problem developed in [11]. The ILP formulation is omitted in this paper due to space constraints, while the results obtained from it are used to evaluate the performance of the distributed algorithm developed here.

### A. Generalized low-point concept

The distributed algorithm for the construction of colored trees to dual homes operates in two phases. The first phase involves distributed depth first search (DFS) numbering of the nodes and computing the generalized low point values, distances, and neighbors. The second phase involves distributed path augmentation for computing the two trees. The DFS numbering and low-point computation helps in identifying paths for augmentation (during the second phase) without backtracking. In order to reduce the average path length, we employ the *generalized low point concept* developed in [1].

Consider a network in which the nodes are numbered in the DFS order. The *low-point value* of a node  $n$  is traditionally defined as the lowest DFS-index of a node that can be reached from  $n$  by using DFS-tree<sup>1</sup> edges and at most one back edge. The *low-point path* of node  $n$  is the path traversed to reach the low-point node. The low-point path of a node  $n$  is of the form  $n \rightarrow i_1 \rightarrow i_2 \rightarrow \dots \rightarrow i_k \rightarrow n'$  ( $k \geq 0$ ) such that: (1) node  $n$  is the DFS-parent of node  $i_1$ , (2) node  $i_{j-1}$  is the DFS-parent

<sup>1</sup>A DFS-tree is a tree rooted at the drain and the arcs in the tree are directed away from the drain. A back edge is an edge that connects a higher DFS-index node to a lower DFS-index node. The *low-point node* of a node  $n$  is the node whose DFS-number is the LPV of node  $n$ .

of node  $i_j$  ( $2 \leq j \leq k$ ), (3) the DFS-index of  $n'$  is lower than that of  $n$ ; and (4) the DFS-index of  $n'$  is the lowest among all such possible paths. The algorithm developed in [14] employs the traditional low-point value and path.

The *generalized low-point value* (GLPV) of a node  $n$  is defined as the lowest DFS-index of a node that can be reached from node  $n$  by traversing a sequence of nodes with increasing DFS-index with the exception of the last hop. The *generalized low-point path* of a node  $n$  is of the form  $n \rightarrow i_1 \rightarrow i_2 \rightarrow \dots \rightarrow i_k \rightarrow n'$  ( $k \geq 0$ ), such that: (1) the DFS-index of  $n$  is lower than that of  $i_1$ , (2) the DFS-index of  $i_{j-1}$  is lower than that of  $i_j$  ( $2 \leq j \leq k$ ), (3) the DFS-index of node  $n'$  is lower than that of node  $n$ , and (4) the DFS-index of  $n'$  is the lowest among all such possible paths. The *generalized low-point neighbor* (GLPN) of a node  $n$  is defined as that neighbor of node  $n$  which is on its generalized low-point path.

### III. LINEAR TIME DISTRIBUTED CONSTRUCTION OF COLORED TREES TO DUAL HOMES

The implementation of the distributed algorithm for the CTDH problem is based on the proof of existence of a solution described in Section II. Assume that we have the graph  $\mathcal{G}'$  obtained from the given network  $\mathcal{G}$  with an additional node  $d$  and two bidirectional links  $(d, d_1)$  and  $(d, d_2)$ . We may employ the distributed algorithm developed for the CT-LD and CT-ND problems with node  $d$  acting as the destination, except that node  $d$  is a “virtual node.”

#### A. Distributed DFS numbering

Assume that the nodes in  $\mathcal{G}'$  are numbered in the DFS order starting from node  $d$ . The DFS-index of node  $d$  is 1 and without loss of generality, assume that the DFS-index of  $d_2$  is 2. In addition, the (generalized) low-point value of  $d_2$  would be 1 and the (generalized) low-point path should traverse node  $d_1$ . As node  $d$  does not exist in the given network, we initialize the DFS-index of  $d_1$  to 1, the DFS-index of  $d_2$  to 2, and allow  $d_2$  to start the DFS numbering phase.

The GLPV and GLPN of a node are computed during the distributed DFS numbering phase. The algorithm to assign the DFS-indices and compute the GLPV and GLPN is shown in Figure 2. The DFS-indices of all the nodes are first initialized to -1. We incorporate hop count as a metric to compute the shortest generalized low-point path among those available. Note that the linear-time algorithm developed in this paper will work with the traditional low-point of a node, however, the path length optimization cannot be made as the arcs are forced to be on the DFS-tree, except the last hop. The node numbers in the example network shown in Figure 1 indicate the DFS-indices of the nodes when numbered from  $d_2$ . The generalized low-point values, distances, neighbors of the nodes are shown in Table I.

#### B. Distributed path augmentation

The distributed path augmentation is based on the generalized path augmentation technique developed in [12]. A generalized version of the same is developed in [13], referred

Notation	Comment
$\text{dfs}[n]$	DFS-index of node $n$ .
$\text{dfsparent}[n]$	DFS-parent of node $n$ .
$\text{glpv}[n]$	Generalized low-point value of node $n$ .
$\text{glpn}[n]$	Generalized low-point neighbor of node $n$ .
$\text{glpd}[n]$	Generalized low-point distance of node $n$ .

DFS(parent, n, currdfs)
1. if $\text{dfs}[n] > 0$ return currdfs;
2. $\text{dfs}[n] = \text{currdfs}$ ; $\text{dfsparent}[n] = \text{parent}$ ; $\text{currdfs} = \text{currdfs} + 1$ ;
3. for every neighbor $i \neq \text{parent}$ of $n$ do:
3.A. $\text{currdfs} = \text{DFS}(n, i, \text{currdfs})$ ;
3.B. if $(\text{dfs}[i] < \text{dfs}[n])$ and $(\text{dfs}[i] \leq \text{glpv}[n])$
3.B.i. $\text{glpv}[n] = \text{dfs}[i]$ ; $\text{glpn}[n] = i$ ; $\text{glpd}[n] = 1$ ;
3.C. else if $(\text{dfs}[i] > \text{dfs}[n])$ and $(\text{glpv}[i] < \text{glpv}[n])$
3.C.i. $\text{glpv}[n] = \text{glpv}[i]$ ; $\text{glpn}[n] = i$ ; $\text{glpd}[n] = \text{glpd}[i] + 1$ ;
3.D. else if $(\text{dfs}[i] > \text{dfs}[n])$ and $(\text{glpv}[i] = \text{glpv}[n])$ and $(\text{glpd}[i] < \text{glpd}[n] - 1)$
3.D.i. $\text{glpn}[n] = i$ ; $\text{glpd}[n] = \text{glpd}[i] + 1$ ;
4. return currdfs;

Fig. 2. Algorithm to assign DFS-indices to the nodes and compute generalized low-point value and neighbor of a node.

TABLE I  
GLPV, GLPN, AND GLPD VALUES OF THE NODES IN THE EXAMPLE NETWORK.

n	glpv(n)	glpn(n)	glpd(n)	n	glpv(n)	glpn(n)	glpd(n)
3	1	4	2	8	1	9	5
4	1	1	1	9	1	10	4
5	1	6	8	10	1	11	3
6	1	7	7	11	1	12	2
7	1	8	6	12	1	1	1

to as the XCT algorithm. The XCT algorithm for the CT-ND or CT-LD problem starts by choosing an arbitrary cycle,  $(d, v_1, \dots, v_k, d)$ , consisting of at least three nodes ( $k \geq 2$ ). The cycle in one direction is added to the red tree and the cycle in the other direction is added to the blue tree. If this cycle does not contain all of the nodes of  $\mathcal{G}$ , then a subsequent path that starts and ends on that cycle and also passes through at least one node not on the cycle is chosen for augmentation. The algorithm continues in this manner until all nodes have been included for augmentation. In order to ensure link-(or node-)disjoint path constraints, a partial ordering among the nodes needs to be maintained. Due to space constraints, we omit the details on the partial ordering and refer the readers to [13] for an elaborate discussion. The distributed algorithm developed in [1] implements partial ordering by maintaining only local (neighborhood) information.

The overview of the path augmentation process is shown in Figure 3. For the CTDH problem, the home with DFS-index of 2 initiates the path augmentation procedure. The path search message, SEARCH, received by a node, is forwarded to

its neighbor based on the forwarding rules described in [1]. The forwarding rules guarantee that the paths are augmented without backtracking, resulting in a linear time algorithm. The neighbors of a node are arranged in the increasing order of their DFS-indices in the neighbor list. Such an arrangement ensures that whenever a path search is initiated from the second home (node with DFS-index 2) through its child, it follows the low-point path, hence resulting in a path to the primary home. Thus, the key difference between the working of the algorithm for the CTDH problem from that of the CT problem for one destination, is that the former finds a path between the two homes while the latter finds a cycle. The arrangement guarantees that the paths from any node to a home will not include the other home, unless the other home is an articulation node<sup>2</sup>.

### Distributed Path Augmentation Algorithm

- 1) Arrange the neighbors in the neighbor list in an increasing order of their DFS indices.
- 2) On receiving a TOKEN message, initiate path search along every node in the neighbor list, one at a time.
- 3) Every node that receives the SEARCH message forwards it sequentially to every node in the neighbor list according to some forwarding rules. If a new path was augmented through a neighbor, then a flag is set corresponding to that node.
- 4) Forward the TOKEN message to every node if the flag is set for that node. The neighbor list is traversed in the *reverse* direction. Every node finishes its operation and sends a RETURN message back.
- 5) After receiving a RETURN message from all the neighbors to whom the token message was sent, send RETURN message to the node that sent the TOKEN message.

Fig. 3. Overview of the steps involved in the distributed algorithm for computing colored trees.

We illustrate the working of the algorithm with the example network shown in Figure 4.

**Step 1.** Node 2 starts the path search through node 3. Based on the forwarding rules [1], the message traverses the generalized low point path until it reaches node 1. Node 1, being a home, is assumed to be a part of the trees, hence sends a SUCCESS message in the reverse direction that the search messages were received. The links that are added to the blue tree are shown in Figure 4(a).

**Step 2.** Node 2 then initiates a path search through node 7. The message traverses the generalized low point path 7–8–9–10–11–12–1. Figure 4(b) shows the path added to the blue tree after this augmentation.

**Step 3.** Now that node 2 has attempted to augment paths through all its neighbors, the TOKEN will be sent to node 7. Node 7 attempts the path through node 6. The path obtained is 7–6–3. Figure 4(c) shows the path added to the blue tree

<sup>2</sup>An articulation node is one which disconnects the network upon its removal.

after this step.

**Step 4.** Node 7 passes the token to node 6 which initiates a search through 5. The path traversed by the message is 6–5–4. Figure 4(d) shows the path added to the blue tree.

The blue and red trees that are generated through this process are shown in Figures 4(d) and 4(e), respectively.

## IV. PERFORMANCE EVALUATION

The linear time distributed algorithm developed in this paper is evaluated on random topologies with 50, 100, 200, and 300 nodes. The topologies were constructed using Waxman’s model [16]. As the solution time of the ILP is prohibitively high for large networks, we compare the results of the distributed algorithm to that of the optimal value for one network topology for each network size. Table II shows the comparison of results obtained from the distributed algorithm employing traditional low point (TLP) and generalized low point (GLP) concepts to the optimal solution obtained by solving the ILP (using CPLEX 8.0 solver [17]). It is observed that the results obtained by employing GLP are closer to the optimal than those obtained by employing TLP. In addition, we observe that the average path length obtained using GLP is at most 12% away from the optimal.

TABLE II

COMPARISON OF RESULTS FROM THE DISTRIBUTED ALGORITHM EMPLOYING TLP AND GLP CONCEPTS TO THE OPTIMAL SOLUTION.

No. of Nodes	No. of Links	Average Red Path Length			Average Blue Path Length		
		TLP	GLP	Optimal	TLP	GLP	Optimal
50	183	6.02	5.53	5.14	4.98	4.2	3.94
100	558	7.47	6.25	5.82	5.63	4.797	4.06
200	1742	15.24	13.81	12.93	7.92	6.96	5.61
300	2129	20.63	17.27	15.1	12.34	7.87	7.35

For each network size, twenty different topologies with two drains were simulated and the average results are shown in Table III for the CTDH-ND case. It is observed that a significant reduction in the average path lengths is obtained by employing the generalized low-point concept, which reduces the hop-count on the low-point path. Similar results were obtained for the CTDH-LD case and are not shown here due to space constraints.

TABLE III

COMPARISON OF RESULTS OBTAINED IN RANDOM NETWORK TOPOLOGIES USING THE DISTRIBUTED ALGORITHM.

No. of Nodes	Average Number of Links	Average Red Path Length		Average Blue Path Length		Average Total Path Length		
		TLP	GLP	TLP	GLP	TLP	GLP	Average Reduction
50	224	4.28	3.53	3.41	2.42	7.69	5.95	22.6%
100	812.2	7.83	6.21	6.71	4.1	14.54	10.31	29.1%
200	1520.8	14.25	11.7	10.37	6.2	24.62	17.9	27.3%
300	2583	20.41	16.07	17.34	8.98	37.75	25.05	32.6%

We evaluate the robustness of the trees constructed using the distributed algorithm under multiple link failures. Let  $H_1$  and  $H_2$  denote the hop length of the two paths from a node to

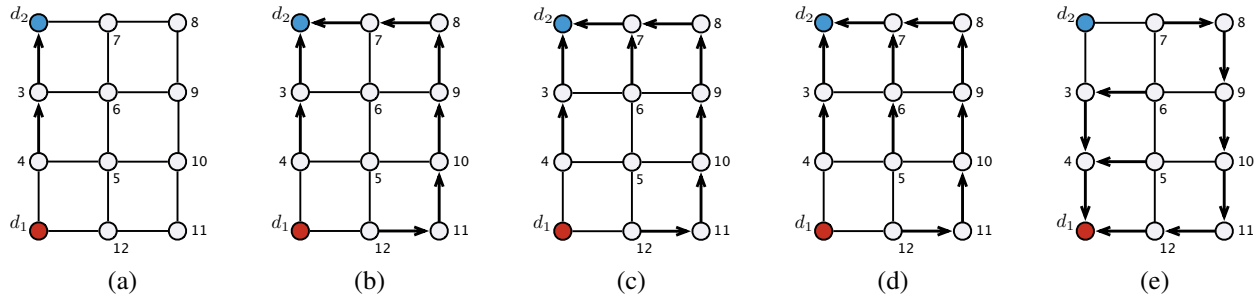


Fig. 4. An example illustrating the stages of the distributed path augmentation technique.

the homes. Given that  $k$  links have failed in the network, the probability that both the paths of a node are affected, denoted by  $P(H_1, H_2, k)$ , is given by:

$$P(H_1, H_2, k) = \sum_{i=1}^{\min(H_1, k-1)} \sum_{j=1}^{\min(H_2, k-1)} \frac{\binom{H_1}{i} \binom{H_2}{j} \binom{L-H_1-H_2}{k-i-j}}{\binom{L}{k}}$$

The sum of the above probability for all the nodes gives the average number of nodes that will lose both the paths to the homes under  $k$ -link failures. Table IV shows the average number of nodes disconnected from both the drains for arbitrary two and three link failures. It is observed that employing GLP reduces the number of nodes disconnected from both the homes in the network in comparison to employing TLP.

TABLE IV

AVERAGE NUMBER OF NODES DISCONNECTED FROM BOTH THE DRAINS FOR ARBITRARY TWO AND THREE LINK FAILURES.

Number of Nodes	Number of Links	Average Number of Nodes Disconnected for $k=2$		Average Number of Nodes Disconnected for $k=3$	
		TLP	GLP	TLP	GLP
50	183	5	4	24	22
100	558	10	7	40	37
200	1742	18	13	74	66
300	2129	21	15	96	87

## V. CONCLUSION

This paper develops a linear-time distributed algorithm for the construction of colored trees in a wireless sensor network employing two drains by modeling the network as a dual-homing network. By allowing the two drains to be roots of the trees, the network is resilient to a single drain failure. This paper also shows that employing the generalized low point (GLP) concept in a DFS tree allows for significant average path length reduction than the traditional low point (TLP) concept. In addition, using GLP rather than TLP allows a network to have greater tolerance for  $k$  link failures in terms of the number of nodes disconnected from both the homes.

## ACKNOWLEDGMENT

The research developed in this paper is supported by National Science Foundation under grants 0325979, 0435490, and EEC-0333046.

## REFERENCES

- [1] S. Ramasubramanian, M. Harkara, and M. Krunz, "Distributed linear time construction of colored trees for disjoint multipath routing," in *Proceedings of IFIP Networking*, Coimbra, Portugal, May 2006, pp. 1026–1038.
- [2] S. Lee and M. Gerla, "Split multipath routing with maximally disjoint paths in ad hoc networks," in *Proceedings of IEEE ICC*, 2001, pp. 3201–3205.
- [3] A. Nasipuri and S. R. Das, "On-demand multipath routing for mobile ad hoc networks," in *Proceedings of IEEE International Conference on Computer Communications and Networks*, October 1999, pp. 64–70.
- [4] J. Wu, "An extended dynamic source routing scheme in ad hoc wireless networks," in *Proceedings of 35th Annual Hawaii International Conference on System Sciences*, January 2002, pp. 3832–3838.
- [5] M. K. Marina and S. R. Das, "On-demand multipath distance vector routing in ad hoc networks," in *Proceedings of IEEE ICNP*, November 2001, pp. 14–23.
- [6] V. D. Park and M. S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *Proceedings of IEEE INFOCOM*, April 1997, pp. 1405–1413.
- [7] J. Raju and J. J. Garcia-Luna-Aceves, "A new approach to on-demand loop-free multipath routing," in *Proceedings of IEEE International Conference on Computer Communications and Networks (ICCCN)*, October 1999, pp. 522–527.
- [8] A. Valera, W. K. G. Seah, and S. V. Rao, "Cooperative packet caching and shortest multipath in mobile adhoc networks," in *Proceedings of IEEE INFOCOM*, March-April 2003, pp. 260–269.
- [9] S. Lee and M. Gerla, "AODV-BR: Backup routing in ad hoc network," in *Proceedings of IEEE WCNC*, September 2000, pp. 1311–1316.
- [10] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *Mobile Computing and Networking*, August 2000, pp. 56–67.
- [11] S. Ramasubramanian, H. Krishnamoorthy, and M. Krunz, "Disjoint multipath routing using colored trees," *Technical Report, University of Arizona*, November 2005.
- [12] M. Medard, R.A. Barry, S.G. Finn, and R.G. Gallager, "Redundant trees for preplanned recovery in arbitrary vertex- redundant or edge redundant graphs," *IEEE/ACM Transactions on Networking*, vol. 7, no. 5, pp. 641–652, October 1999.
- [13] G. Xue, L. Chen, and K. Thulasiraman, "Quality-of-service and quality-of-protection issues in preplanned recovery schemes using redundant-trees," *IEEE Journal on Selected Areas in Communication*, vol. 21, no. 8, pp. 1332–1345, October 2003.
- [14] W. Zhang, G. Xue, J. Tang, and K. Thulasiraman, "Linear time construction of redundant trees for recovery schemes enhancing QoP and QoS," in *Proceedings of IEEE INFOCOM*, Miami, FL, USA, March 2005, pp. 2702–2710.
- [15] J. Wang, V.M. Vokkarane, R. Jothi, X. Qi, B. Raghavachari, and J.P. Jue, "Dual homing protection in IP-over-WDM networks," *Journal of Lightwave Technology*, vol. 23, no. 10, pp. 3111–3119, 2005.
- [16] B. M. Waxman, "Routing of multipoint connections," *IEEE Journal of Selected Areas in Communications*, vol. 6, no. 9, pp. 1617–1622, December 1988.
- [17] CPLEX Solver, <http://www.cplex.com>.