# CONSTRUCTION AND EVALUATION OF MESHES BASED ON SHORTEST PATH TREE VS. STEINER TREE FOR MULTICAST ROUTING IN MOBILE AD HOC NETWORKS

**[1]JAMES SIMS, [2]NATARAJAN MEGHANATHAN**

[1]Undergrad Student, Department of Computer Science, Jackson State University, Jackson, MS 39217, USA

[2]Assistant Professor, Department of Computer Science, Jackson State University, Jackson, MS 39217, USA

**ABSTRACT**

A mobile ad hoc network (MANET) is a network of mobile devices that continuously restructure their topology due to mobility. Proposals on tree-based routing versus mesh-based routing protocols have shown that mesh-based multicast routing gives improved results for MANETs. The main reason for this is because trees are highly susceptible to failure due to frequent mobility. This paper will look at the developing a mesh from two different structures: a shortest path tree and a minimum Steiner tree. In each of the two cases, we extend a tree to a mesh by incorporating edges, which exist in the network graph, between any two constituent nodes of the tree. The goal is to look and determine how the implementation of these two structures can affect the overall performance of the multicast mesh and derive theoretical

**Keywords:** *Mobile Ad hoc Networks, Shortest Path Trees, Minimum Steiner Trees, Stability, Multicast, Mesh*

## 1. INTRODUCTION

The study of mobile ad hoc networks (MANETs) continues to grow mainly because of the variety of applications that these networks can be applied to: disaster recovery, rescue missions, military operations in a battlefield, conferences, crowd control, etc [1]. MANETs are of top priority in these areas where a few seconds or loss of communication, could be the difference between losing and saving lives. It has been well documented in many papers that in the case of MANETs, mesh-based routing protocols perform better than tree-based routing protocols for multicasting [2]. This fact leads us into our research to look into how different implementations of the mesh can either help or hurt the performance of the mesh in its initial phase. In this paper, we will focus on the structure of the mesh and its formation, and not focus on the transmission of data between nodes, overhead, and data packet delivery ratio. We will take a different approach to form a mesh – we will extend a tree (shortest path tree and a minimum Steiner tree) to a mesh by adding edges that may exist, in the original network graph, between any two constituent nodes of the tree. The goal is to observe if the mesh becomes more efficient, specifically, in terms of its stability, measured as the lifetime of the mesh. In addition, we will also measure the hop count per source-destination pair in a mesh as well as the number of edges that constitutes the mesh.

The main areas that are being researched on MANETs are targeting performance with respect to stability, energy consumption and control message overhead. By using a mesh structure instead of a tree, the disadvantages of multicast trees in mobile wireless networks are avoided [4]. In this study, the networks will be constructed as a unit-disk graph where all of the edges of the graph are bi-directional. These properties will allow us develop a minimum Steiner tree using Kou et al's Heuristic [5] to find and approximate minimum Steiner tree, and also find the minimal weight per source-receiver path for a shortest path tree. The mesh can then be developed after the trees have been constructed by using the criteria of links that may exist between the nodes of the tree. We believe that the two different implementations (shortest path tree based mesh vs. Steiner tree based mesh) can have a major effect on the performance of the multicast group mesh with respect to stability, delay, energy consumption and other critical metrics.

The rest of the paper is organized as follows. Section 2 describes the construction of a mesh from a shortest path tree; Section 3 describes the construction of a mesh based on a minimum Steiner tree. Section 4 describes the simulation conditions, presents and interprets the performance results. Section 5 concludes the paper and also lists ideas for future work. Throughout the paper, the terms 'edge' and 'link', 'node' and 'vertex', 'path' and 'route' are used interchangeably. They mean the same.

## 2. SHORTEST PATH TREE-BASED MESH

In order to find a shortest path tree from the source node(s) to the receiver node(s) constituting the multicast group, the underlying network graph must first be connected. This means there should be a path between any two nodes in the network graph. We will use the well-known Breadth First Search (BFS) algorithm [6] to test the connectivity of the network graph. If the network is connected at the specific time instant we want to find the tree, we proceed to find the shortest path tree rooted at each of the source nodes to the set of receiver nodes. We will use the well-known Dijkstra's shortest path algorithm [6] to determine the shortest path trees from a source node to the receiver nodes in a given graph. The pseudo code for Dijkstra's algorithm is shown here in Figure 1 [6]. We will extend this tree to a mesh by checking if there are edge(s) between any two constituent nodes (the source node, intermediate nodes and receiver nodes) of the shortest path tree. The end result is a shortest path tree overlapped with mesh edges, resulting in a multicast mesh that has more robustness for node mobility, compared to a tree.

```
Begin Algorithm Dijkstra (G, s)
1    For each vertex v Є V
2        d [v] ← ∞ // an estimate of the minimum
                    -weight path from s to v
3    End For
4    d [s] ← 0
5    S ← Φ  // set of nodes for which we know the
                minimum-weight path from s
6    Q ←  V // set of nodes for which we know
            estimate of minimum-weight path from s
7    While Q ≠ Φ
8        u ← EXTRACT-MIN (Q)
9        S ← S U {u}
10       For each vertex v such that (u, v) Є E
11           If d [v] > d [u] + w (u, v) then
12               d [v] ← d [u] + w (u, v)
13               Predecessor (v) = u
13           End If
14       End For
15   End While
16 End Dijkstra
```

**Figure 1:** *Pseudo Code for Dijkstra's Algorithm*

We illustrate the extension of a shortest path tree to a shortest path mesh through the examples in Figures 2 and 3. In Figure 2, we have one source (node F) and three receiver nodes (nodes L, C and N). The shortest path tree determined through Dijkstra algorithm will have 7 links. In addition, we notice that there exist links between nodes M-W, W-K, L-E, W-N and E-N wherein M, W, K, L, E and N are constituent nodes of the tree. As these nodes are part of the shortest path tree, it is prudent to extend the fragile tree (that can break anytime with the failure of a single link) to a more robust mesh (that can withstand link failures) by adding the links that exist between the constituent nodes of the tree. In our case, the above listed five links along with the seven links of the shortest path tree form a robust mesh involving 12 links (bold lines), comprising the same set of nodes that formed the shortest path tree. Note that we could not include any link connected to nodes A and S to the extended mesh as these two nodes are not part of the original shortest path tree.

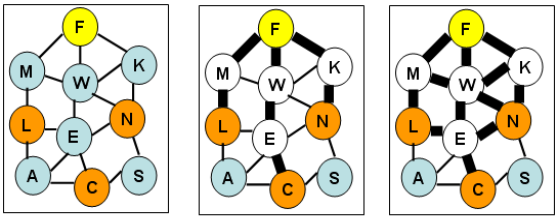


**Figure 2:** *Extension of a Single Source Shortest Path Tree to a Mesh*

We will also apply the above idea for multi-source scenarios, as illustrated through the example in Figure 3. In this case, we run the Dijkstra algorithm to find the shortest path tree rooted at each source node (nodes A and C) connecting the set of receiver nodes (nodes F, M and O). We then form an aggregate of all the shortest path trees rooted at each source node. Such an aggregate of shortest path trees has 13 links. We could extend the shortest path tree aggregate to a 19-link mesh by incorporating additional links between the

constituent nodes of the original shortest path trees. Note that, we could not add links connected to nodes D, H, L and P as these four nodes are not the constituent nodes of the original aggregate of shortest path trees.

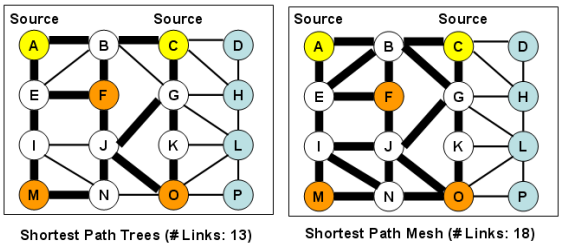


**Figure 3:** *Extension of a Two Source Shortest Path Tree Aggregate to a Mesh*

## 3. MINIMUM STEINER TREE-BASED MESH

Given a static graph, $G = (V, E)$, where $V$ is the set of vertices, $E$ is the set of edges and a subset of vertices (called the multicast group or Steiner points) $MG \subseteq V$, the multicast Steiner tree is the tree with the least number of edges required to connect all the vertices in $MG$. Unfortunately, the problem of determining a minimum edge Steiner tree in an undirected graph like that of the static graph is NP-complete. Efficient heuristics (e.g., [5]) have been proposed in the literature to approximate a minimum Steiner tree. In this paper, we use the Kou et al's [5] well-known $O(|V||MG|^2)$ heuristic ($|V|$ is the number of nodes in the network graph and $|MG|$ is the size of the multicast group comprising of the source nodes and the receiver nodes) to approximate the minimum edge Steiner tree in graphs representing snapshots of the network topology. An *MG-Steiner-tree* is referred to as the minimum edge Steiner tree connecting the set of nodes in the multicast group $MG \subseteq V$. In unit disk graphs such as the static graphs used in our research, Step 5 of the heuristic is not needed and the minimal spanning tree $T_{MG}$ obtained at the end of Step 4 could be considered as the minimum edge Steiner tree. We use the Kruskal's algorithm [6] to determine the minimum spanning trees.

**Input:** A Static Graph $G = (V, E)$
Multicast Group $MG \subseteq V$

**Output:** A *MG-Steiner-tree* for the set $MG \subseteq V$

**Begin** Kou et al Heuristic ($G$, $MG$)

**Step 1:** Construct a complete undirected weighted graph $G_C = (MG, E_C)$ from $G$ and $MG$ where $\forall (v_i, v_j) \in E_C$, $v_i$ and $v_j$ are in $MG$, and the weight of edge $(v_i, v_j)$ is the length of the shortest path from $v_i$ to $v_j$ in $G$.

**Step 2:** Find the minimum weight spanning tree $T_C$ in $G_C$ (If more than one minimal spanning tree exists, pick an arbitrary one).

**Step 3:** Construct the sub graph $G_{MG}$ of $G$, by replacing each edge in $T_C$ with the corresponding shortest path from $G$ (If there is more than one shortest path between two given vertices, pick an arbitrary one).

**Step 4:** Find the minimal spanning tree $T_{MG}$ in $G_{MG}$ (If more than one minimal spanning tree exists, pick an arbitrary one). Note that each edge in $G_{MG}$ has weight 1.

**return** $T_{MG}$ as the *MG-Steiner-tree*

**End** Kou et al Heuristic

**Figure 4:** *Kou et al's Heuristic [5] to find an Approximate Minimum Edge Steiner Tree*

We give a brief outline of the heuristic in Figure 4 and illustrate the working of the heuristic through an example in Figure 5. The vertices {D, G, E, M, N, P} form the multicast group in the vertex set {A, B … P}. As observed in the example, the subgraph $G_{MG}$ obtained in Step 3 is nothing but the minimal spanning tree $T_{MG}$, which is the output of Step 4. In general, for unit disk graphs, like the static graphs we are working with, the outputs of both Steps 3 and 4 are the same and it is enough that we stop at Step 3 and output the MG-Steiner-tree.

The multicast mesh based on the minimum Steiner tree is constructed the same as the shortest path tree-based mesh. We check for the existence of edge(s) between any two constituent nodes of the minimum Steiner tree for its extension to a mesh. Figure 5 also illustrates the extension of the minimum Steiner tree to a mesh. As we notice in this example, the number of edges in the Steiner tree-based mesh is just one more than the number of edges in the original minimum Steiner tree. In general, as we notice in the simulation results too, the number of edges that could be added to a Steiner-tree based mesh from its corresponding Steiner tree is relatively lower than the number of edges that could be added to a shortest path tree based mesh from its corresponding tree (i.e., the shortest path tree). As the shortest path trees have more edges than the Steiner trees, the possibility of adding relatively more edges lends more robustness to the shortest path tree based meshes compared to the Steiner-tree based meshes. This crucial

observation, confirmed through the simulation results, is one of the major contributions of this paper.

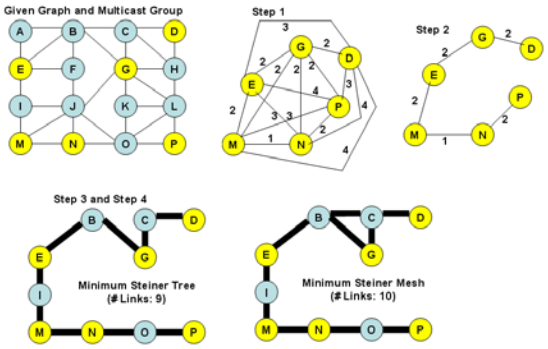


**Figure 5:** *Construction of a Minimum Steiner Tree and its Extension to a Mesh*

## 4. SIMULATIONS AND RESULTS

The simulations have been conducted in a discrete-event simulator implemented by the authors in Java. The two multicast mesh algorithms (shortest path based mesh, denoted as SPT mesh in the figures and Steiner-tree based mesh) have been implemented in a centralized fashion. The network size is 1000 m x 1000 m. The density of the network is varied by conducting the simulations with two different values for the number of nodes: 50 nodes (low density) and 150 nodes (high density). The transmission range per node is 250m. The simulation time is 1000 seconds. The network is periodically sampled for every 0.25 seconds and on such a snapshot network graph, we validate the existence of the most recently used multicast mesh or determine new multicast mesh, if no such mesh exists. A multicast mesh is used as long as it exists. We used three different values for the number of source nodes: 1, 5 and 10 sources; the number of receivers per source (same set of receiver nodes for all sources) is varied with five different values: 3, 10, 18, 27 and 36.

The node mobility model used is the Random Waypoint model [7]. Each node starts moving from an arbitrary location (i.e., waypoint) at a speed uniformly distributed in the range [$v_{min}$, …, $v_{max}$]. Once the destination is reached, the node may stop there for a certain time called the pause time and then continue to move to a new waypoint by choosing a different target location and a different velocity. A mobility trace file generated for a particular $v_{max}$ value over the duration of the simulation time is the congregate of the location, velocity and time information of all the waypoints for every node in the network. In this paper, we set $v_{min}$ = 0. The $v_{max}$ values used are 5 m/s (low mobility) and 50 m/s (high mobility). The pause time is 0 seconds.

### 4.1 Performance Metrics

The performance metrics measured are as follows. Each performance metric illustrated in Figures 6 to 17 is an average of the values measured using 5 different lists of receiver nodes of a particular size, for a given number of source nodes and the multicast mesh algorithm is run on five different mobility trace files generated for a fixed $v_{max}$.

(i) *Lifetime per Mesh*: Whenever a path does not exist from any source to all receivers, the mesh is considered to have failed. A new mesh is determined after the network graph is found to be connected for the time instant. The lifetime of the mesh is the average amount of time these meshes exist the duration of the simulation.

(ii) *Edges per Mesh*: The number of edges per mesh is the time-averaged value of the number of edges in the sequence of meshes used for each of the time instants of the multicast session. For each time instant, each edge in a mesh is validated for its existence before the validity check for mesh connectivity. The concept of time-average is explained as follows using an example: If we have been using two meshes – the first mesh with 20 edges for 9 seconds and the second mesh with 15 edges for the subsequent 6 seconds – over a time period of 15 seconds, the time-averaged value for the number of edges per mesh is (20*9 + 15*6)/15 = 18.0 and not simply the average of the two values for the number of edges: (20 + 18) / 2 = 17.5.

(iii) *Hop Count per Source-Receiver Path*: This metric is a time-averaged value of the number of hops in the paths from each source to each receiver of the multicast group for the duration of the multicast session.

### 4.2 Lifetime per Mesh

The average lifetime per mesh is a measure of the stability of the mesh. The lifetime of both the shortest path tree-based mesh and minimum Steiner tree-based mesh increased as network density increased. The lifetime of the shortest path tree-based mesh was significantly better than that of the minimum Steiner tree-based mesh. Only in cases of low mobility, low network density, and smaller multicast group size was the minimum Steiner tree mesh lifetime in the range of the shortest path tree

mesh. In over 90% of the cases the average lifetime for the shortest path tree mesh was at least twice the amount of time greater than that of the minimum Steiner mesh lifetime.

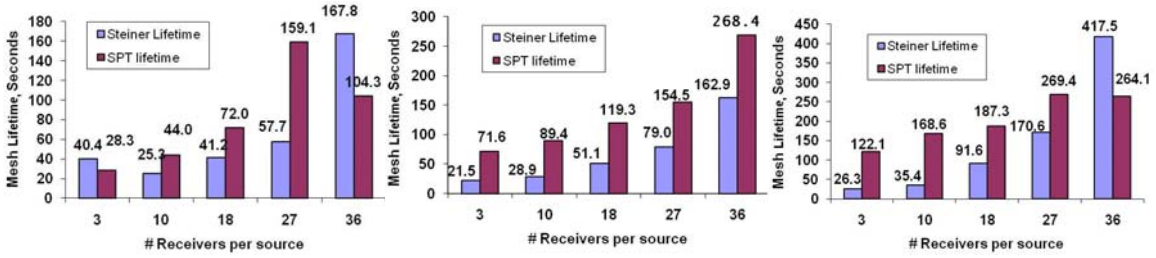


**Figure 6.1:** 1 *source* **Figure 6.2:** 5 *sources* **Figure 6.3:** 10 *sources*

**Figure 6:** *Lifetime per Mesh (Low Node Mobility and Low Network Density)*

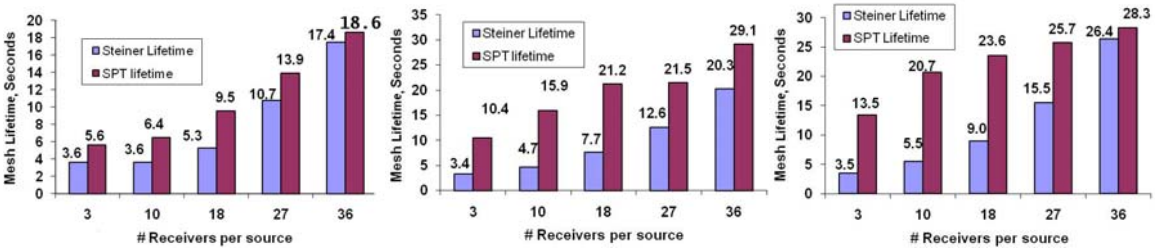


**Figure 7.1:** 1 *source* **Figure 7.2:** 5 *sources* **Figure 7.3:** 10 *sources*

**Figure 7:** *Lifetime per Mesh (High Node Mobility and Low Network Density)*

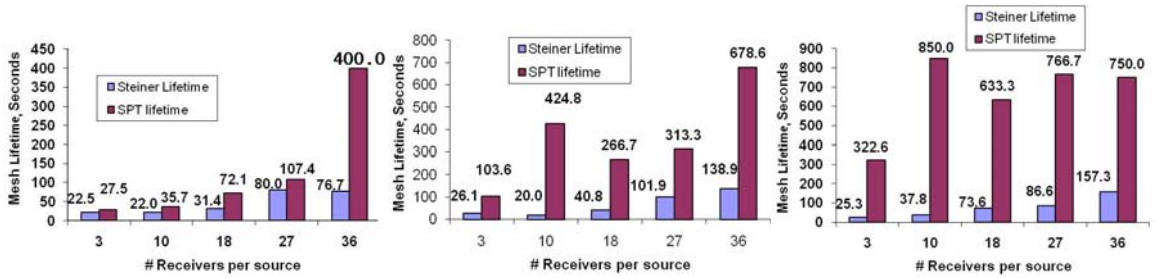


**Figure 8.1:** 1 *source* **Figure 8.2:** 5 *sources* **Figure 8.3:** 10 *sources*

**Figure 8:** *Lifetime per Mesh (Low Node Mobility and High Network Density)*

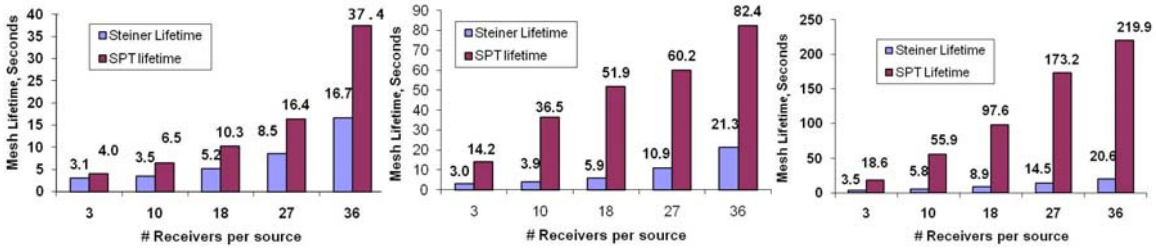


**Figure 9.1:** 1 *source* **Figure 9.2:** 5 *sources* **Figure 9.3:** 10 *sources*

**Figure 9:** *Lifetime per Mesh (High Node Mobility and High Network Density)*

When the multicast group size (sum of the number of sources plus the number of receivers) exceeded 10, the average lifetime per shortest path mesh was more than 200 seconds (i.e., more than 20% of the 1000s simulation time). When node mobility was 5 m/s and network density was 150 nodes, the average lifetime per shortest path mesh was almost 60% of the simulation time. As the node velocity increased, the mesh lifetime decreased as expected.

### 4.3 Number of Edges per Mesh

The shortest path mesh had a significantly larger number of edges and this could be attributed to the presence of a relatively larger number of nodes per mesh as well as to the principle of the underlying

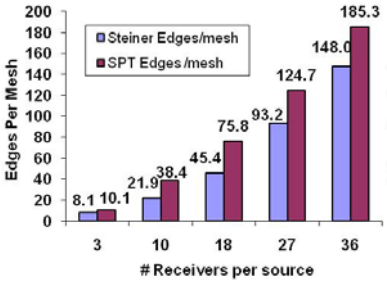


**Figure 10.1:** 1 *source*

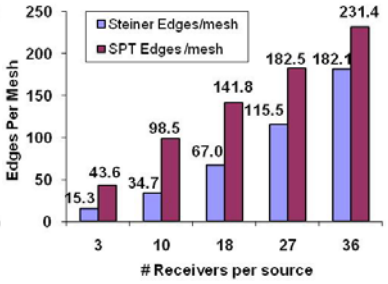


**Figure 10.2:** 5 *sources*

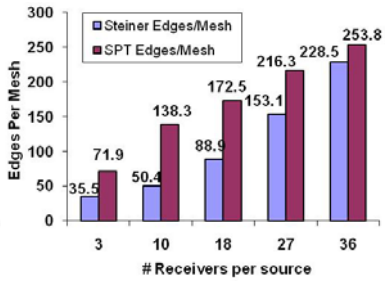


**Figure 10.3:** 10 *sources*

**Figure 10:** *Number of Edges per Mesh (Low Node Mobility and Low Network Density)*

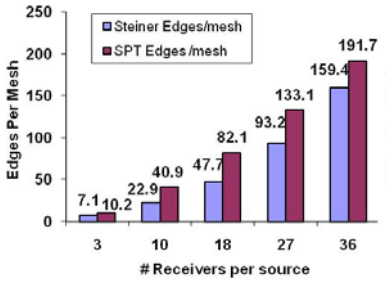


**Figure 11.1:** 1 *source*

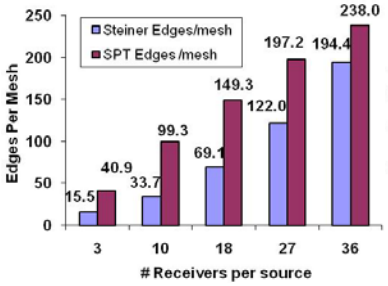


**Figure 11.2:** 5 *sources*

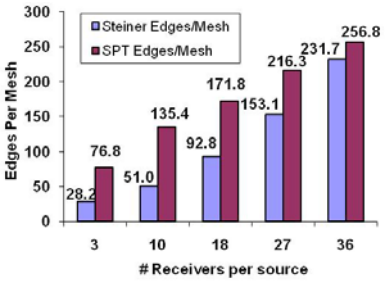


**Figure 11.3:** 10 *sources*

**Figure 11:** *Number of Edges per Mesh (High Node Mobility and Low Network Density)*

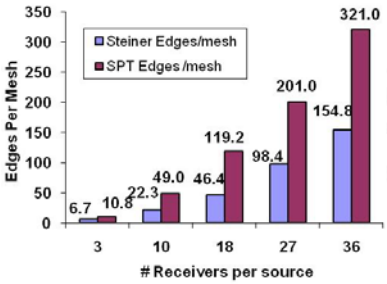


**Figure 12.1:** 1 *source*

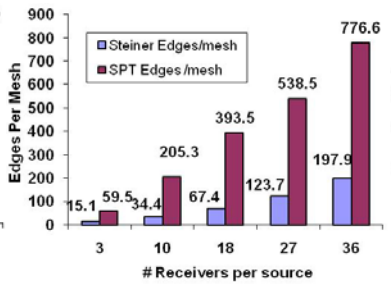


**Figure 12.2:** 5 *sources*

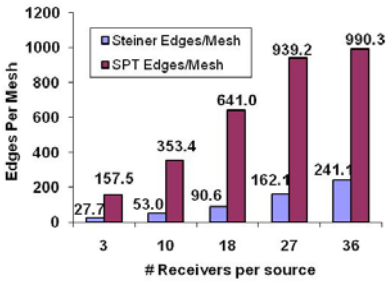


**Figure 12.3:** 10 *sources*

**Figure 12:** *Number of Edges per Mesh (Low Node Mobility and High Network Density)*

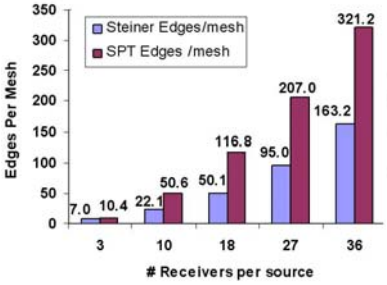


**Figure 13.1:** 1 *source*

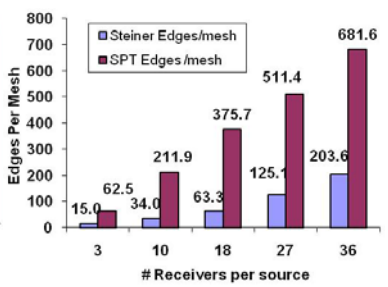


**Figure 13.2:** 5 *sources*

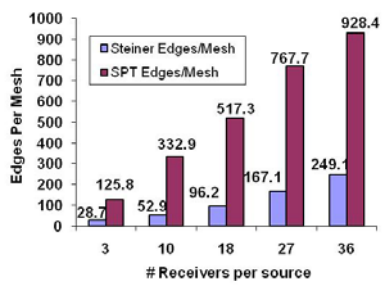


**Figure 13.3:** 10 *sources*

**Figure 13:** *Number of Edges per Mesh (High Node Mobility and High Network Density)*

shortest path tree algorithm in determining shortest minimum hop paths without any consideration on the number of edges being introduced to the tree. Only, in conditions of low node mobility and low network density, both the meshes had similar values for the number of nodes. The number of nodes per shortest path mesh grew to 1.5-2.0 times larger than the number of nodes per Steiner tree mesh as the network density increased. As a result, for at least 70% of the scenarios, the shortest path

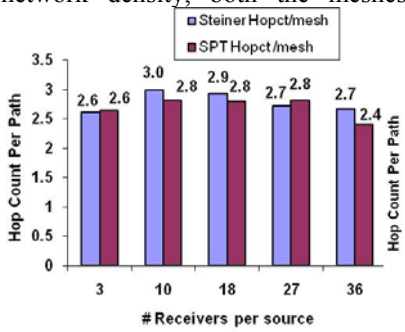


**Figure 14.1:** 1 *source*

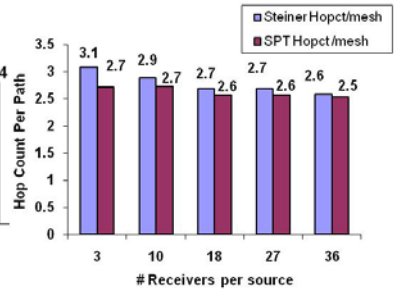


**Figure 14.2:** 5 *sources*

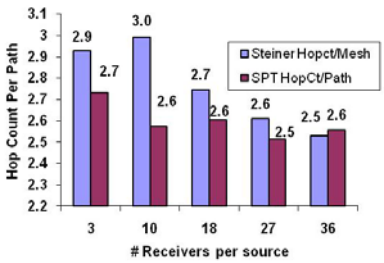


**Figure 14.3:** 10 *sources*

**Figure 14:** *Hop Count per Source-Receiver Pair Path (Low Node Mobility and Low Network Density)*

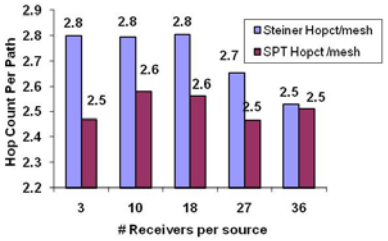


**Figure 15.1:** 1 *source*

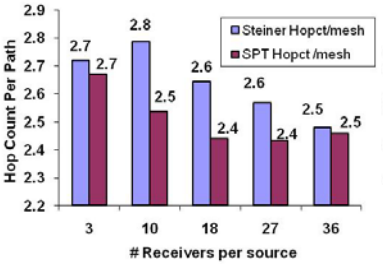


**Figure 14.2:** 5 *sources*

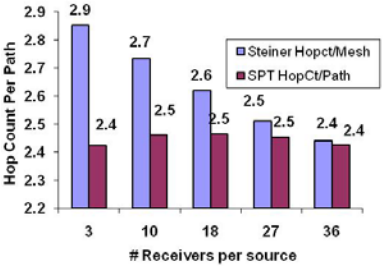


**Figure 14.3:** 10 *sources*

**Figure 15:** *Hop Count per Source-Receiver Pair Path (High Node Mobility and Low Network Density)*

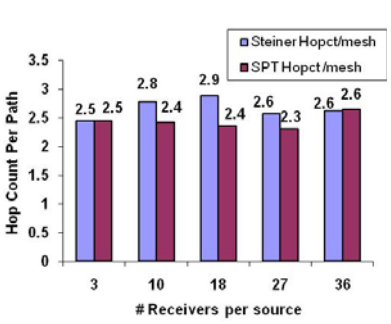


**Figure 16.1:** 1 *source*

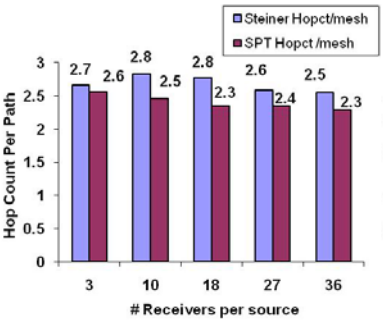


**Figure 16.2:** 5 *sources*

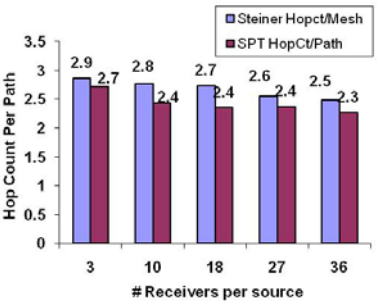


**Figure 16.3:** 10 *sources*

**Figure 16:** *Hop Count per Source-Receiver Pair Path (Low Node Mobility and High Network Density)*

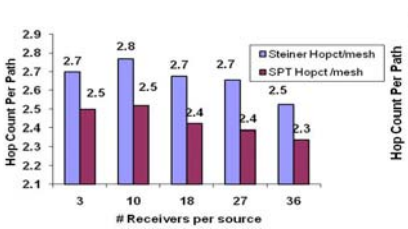


**Figure 16.1:** 1 *source*

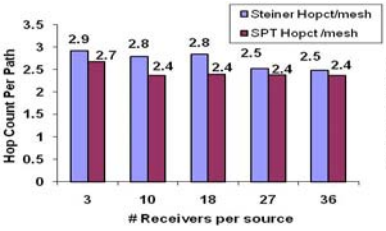


**Figure 16.2:** 5 *sources*

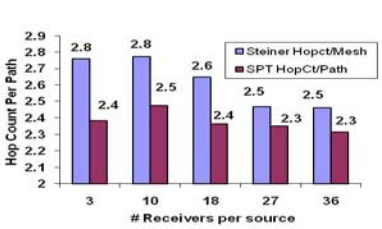


**Figure 16.3:** 10 *sources*

**Figure 17:** *Hop Count per Source-Receiver Pair Path (High Node Mobility and High Network Density)*

mesh had at least twice the number of edges compared to the Steiner tree-based mesh.

The mesh lifetime is directly related to the number of nodes and edges per mesh. Having more intermediate nodes and edges in the mesh lends robustness to the mesh. The tradeoff is a relatively larger routing overhead. There would be several paths for every source-receiver pair and packets from the source nodes go through all these paths to the receiver nodes. Hence, there would be lot of redundant packets received along a shortest path mesh vis-à-vis a Steiner-tree based mesh.

### 4.4 Hop Count per Source-Receiver Path

The average hop count per source-receiver path for both the shortest path tree and minimum Steiner tree-based meshes was in the range from 2-3.5 hops; the shortest path mesh had a lower hop count compared to the Steiner-tree mesh for more than 85% of the scenarios. This could be attributed to the operating principle of the underlying shortest path tree algorithm as well as to the presence of a larger number of edges per mesh. The Steiner-tree based meshes are based on the objective of minimizing the number of edges per tree to reduce the route redundancy. The tradeoff is that when the mesh nodes are connected with less number of edges, the paths between the source nodes to the receiver nodes are more likely to be longer. With a shortest path tree based mesh, every source-receiver pair (even in multi-source scenarios) is more likely to be connected on a relatively shorter path compared to the Steiner-tree based mesh. We also noted that the average hop count per path was not much affected either by the network density, node mobility, or the number of source nodes and receiver nodes of the multicast session.

## 5. CONCLUSIONS AND FUTURE WORK

The stability of any communication structure depends heavily on the design and operating principle of its base, and this is the approach we took to design and observe tree-based meshes (shortest path mesh vis-à-vis a Steiner-tree based mesh) for a MANET. We observe that the shortest path meshes had a longer lifetime and lower hop count per source-receiver path and this could be attributed to the presence of a relatively larger number of nodes and edges per shortest path mesh. In a related work [1], we observed that the shortest path (minimum hop) trees are less stable the Steiner (minimum edge) trees. However, as we noted in this research, the stability of the meshes that are extended from the shortest path trees could be considerably high compared to the stability of the meshes that are extended from the Steiner trees. In other words, the shortest path trees may be more fragile than Steiner trees; but, the shortest path tree based meshes are more robust than the Steiner-tree based meshes. The tradeoff for a larger mesh stability would be the redundancy in the number of similar data packets that get transferred through the mesh. The algorithm to construct Steiner-tree based meshes also has a larger run-time compared to that of the shortest path tree-based meshes.

We currently plan to develop distributed versions of the two (centralized) mesh algorithms that we have worked on this paper so that they could be evaluated with respect to more practical performance metrics such as energy consumption, delay per data packet and packet delivery ratio. We will also investigate techniques that will reduce the redundancy in the data packets received at the multicast receivers through the mesh, without any compromise on the robustness of the mesh.

## 6. ACKNOWLEDGMENTS

This research is funded by the U.S. National Science Foundation through grant (CNS-0851646) entitled: “REU Site: Undergraduate Research Program in Wireless Ad hoc Networks and Sensor Networks,” hosted by the Department of Computer Science at Jackson State University (JSU), MS, USA. The authors also acknowledge Dr. M. Watts, Dr. L. Moore, Mrs. B. Johnson and Ms. I. Dasari for their services during the Summer 2010 program.

## REFERENCES:

[1] N. Meghanathan, “Determining a Sequence of Stable Multicast Steiner Trees in Mobile Ad hoc Networks,” *Proceedings of the 44th ACM Southeast Conference*, pp. 102-106, Melbourne, FL, USA, March 2006.

[2] K. Viswanath, K. Obraczka and G. Tsudik, “Exploring Mesh and Tree-based Multicast Routing Protocols for MANETs,” *IEEE Transactions on Mobile Computing*, vol. 5, no. 1, pp. 28-42, January 2006.

[3] M. Lee and Y. Kim, “PatchODMRP: an ad hoc multicast routing protocol,” *Proceedings of the 15th International Conference on Information Networking*, pp. 537-543, February 2001.

[4] S-J. Lee, M. Gerla and C-C. Chiang, "On-Demand Multicast Routing Protocol," *Proceedings of the IEEE Wireless Communications and Networking Conference*, vol. 3, pp. 1298-1302, September 1999.

[5] L. Kou, G. Markowsky and L. Berman, "A Fast Algorithm for Steiner Trees," *Acta Informatica*, vol. 15, no. 2, pp. 141-145, 1981.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, "Introduction to Algorithms," 2nd Edition, pp. 561-579, MIT Press, 2001.

[7] Bettstetter, C., Hartenstein, H., and Perez-Costa, X. "Stochastic Properties of the Random-Way Point Mobility Model," *Wireless Networks*, vol. 10, no. 5, pp. 555-567, September 2004.