Benchmarks and Tradeoffs for Minimum Hop, Minimum Edge and Maximum Lifetime per Multicast Tree in Mobile Ad hoc Networks

Natarajan Meghanathan
Department of Computer Science
Jackson State University
Jackson, MS 39217, USA
natarajan.meghanathan@jsums.edu

Abstract

The high-level contribution of this paper is to establish benchmarks for the minimum hop count per source-receiver path, minimum number of edges per tree and maximum tree lifetime for multicast routing in mobile ad hoc networks (MANETs) and explore the tradeoffs between these three metrics. Accordingly, we consider three categories of algorithms – Breadth First Search (for minimum hop trees), minimum Steiner tree heuristic (for minimum edge trees) and the recently proposed *OptTreeTrans* algorithm (for maximum lifetime trees). Extensive simulations of the above three algorithms on centralized snapshots of the MANET topology, sampled for every 0.25 seconds, have been conducted for 1000 seconds under two different conditions of network density and three different multicast group sizes. Simulation results illustrate that minimum edge trees have 20-160% larger lifetime than the minimum hop trees; but still, the minimum edge trees have only 6-14% of the maximum tree lifetime possible. The tradeoff is that the minimum edge trees and maximum lifetime trees respectively have 20-100% and 28-86% larger hop count per source-receiver path compared to the minimum hop trees. Similarly, the minimum hop trees and maximum lifetime trees respectively have 13-35% and 35-68% more edges than the minimum edge trees. Thus, the above three performance metrics cannot be simultaneously optimized and MANET multicast routing can be envisioned to have at least three mutually contrasting categories of protocols.

Keywords: *Minimum Hop, Minimum Edge, Maximum Lifetime, Multicast Routing, Mobile Ad hoc Networks, Simulations, Steiner Trees*

1. Introduction

A mobile ad hoc network (MANET) is a dynamic distributed system of wireless nodes that move independent of each other in an autonomous fashion. The network bandwidth is limited and the medium is shared. As a result, transmissions are prone to interference and collisions. The battery power of the nodes is constrained and hence nodes operate with a limited transmission range, often leading to multi-hop routes between any pair of nodes in the network. Communication structures (e.g.., paths, trees, connected dominating sets and etc) for routing in wireless ad hoc networks could be principally based on two different approaches [1]: Optimum Routing Approach (ORA) and Least Overhead Routing Approach (LORA). With ORA, the communication structure used at any time instant is always the optimum with respect to a particular metric. On the other hand, with LORA, a communication structure determined for optimality with respect to a particular metric at a time instant is used in the subsequent time instants as long as the communication structure exists. For dynamically changing, distributed, resource-constrained MANETs, the LORA strategy is often preferred over the ORA strategy to avoid the communication overhead incurred in determining the optimum communication structure at every time instant. Hence, we focus on using LORA for the rest of this paper.

Multicasting in ad hoc wireless networks has numerous applications in collaborative and distributed computing like civilian operations (audio/ video conferencing, corporate communications, distance learning, outdoor entertainment activities), emergency search-and-rescue, law enforcement and warfare situations, where establishing and maintaining a communication infrastructure may be expensive or difficult. A common feature among all these applications is one-to-many and many-to-many communications among the participants [2].

Several MANET multicast routing protocols have been proposed in the literature [1]. They are mainly classified as: tree-based and mesh-based protocols. In tree-based protocols, only one route exists between a source and a destination and hence these protocols are efficient in terms of the number of link transmissions. The tree-based protocols can be further divided into two types: source tree-based and shared tree-based. In source tree-based multicast protocols, the tree is rooted at the source. In shared tree-based multicast protocols, the tree is rooted at a core node and all communication between the multicast source and the receiver nodes is through the core node. Even though shared tree-based multicast protocols are more scalable with respect to the number of sources, these protocols suffer under a single point of failure, the core node. On the other hand, source tree-based protocols are more efficient in terms of traffic distribution. In mesh-based multicast protocols, multiple routes exist between a source and each of the receivers of the multicast group. A receiver node receives several copies of the data packets, one copy through each of the multiple paths. Mesh-based protocols provide robustness at the expense of a larger number of link transmissions leading to inefficient bandwidth usage. Considering all the pros and cons of these different classes of multicast routing protocols for MANETs, we feel the source tree-based multicast routing protocols are more efficient in terms of traffic distribution and link usage.

Not much work has been done towards the evaluation of MANET multicast routing from a theoretical point of view with respect to metrics such as the hop count per source-receiver path, number of edges per multicast tree and the lifetime per tree. These three theoretical metrics significantly contribute and influence the more practically measured performance metrics such as the energy consumption per node, end-to-end delay per data packet, multicast routing overhead and etc. that have been often used to evaluate and compare the different MANET multicast routing protocols in the literature. Hence, we take a different approach in this paper. We study MANET multicast routing using the theoretical algorithms that would yield the benchmarks (i.e., optimum values) for the above metrics – the Breadth First Search (BFS) algorithm [3] for the minimum hop count per source-receiver path; the minimum Steiner tree heuristic [4] for the minimum number of edges and the recently proposed algorithm for the optimal number of tree transitions (*OptTreeTrans* algorithm) [5] for multicast trees with the maximum lifetime.

Our simulation methodology is outlined as follows: Using the mobility profiles of the nodes gathered offline from a discrete-event simulator (ns-2 [6]), we will generate snapshots of the MANET topology, referred to as Static Graphs, periodically for every fixed time instant. For simulations with a particular algorithm, if a multicast tree is not known for a particular time instant, we will run the algorithm on the static graph in a centralized fashion and adopt the LORA strategy of using this multicast tree as long as it exists for the subsequent static graphs. If the tree no longer exists after a certain time instant, the multicast algorithm is again run to determine a new tree. This procedure is repeated for the entire simulation time. Depending on the algorithm used, the sequence of multicast trees generated either have the minimum hop count per source-receiver path, minimum number of edges or the maximum lifetime. Our hypothesis is that the multicast trees, determined to optimize one of the three theoretical metrics, would be sub-optimal with respect to the other two metrics. Through extensive simulation analysis, we confirm our hypothesis to be true and we explain in detail the performance tradeoffs associated with the three theoretical metrics.

The rest of the paper is organized as follows: Section 2 introduces the notion of a Static Graph and a Mobile Graph; and reviews the BFS algorithm, Kou et al.'s heuristic for minimum Steiner trees and the recently proposed *OptTreeTrans* algorithm for the optimal number of tree transitions. Section 3 presents the simulation results for the benchmark values of the three theoretical metrics and simultaneously explores the tradeoffs between these metrics. Section 4 concludes the paper. For the rest of the paper, the terms 'vertex' and 'node', 'algorithm' and 'heuristic', 'destination' and 'receiver', 'link' and 'edge' are used interchangeably. They mean the same.

2. Review of the Theoretical Algorithms used for Multicast Simulations

In this section, we first describe the notion of static graphs and mobile graph, referring to the snapshots of the network topology, on which we run the theoretical algorithms to simulate multicasting. We then describe the three algorithms (BFS, Minimum Steiner tree and *OptTreeTrans*) used in this paper.

2.1 Static Graph and Mobile Graph

A static graph is a snapshot of the MANET topology at a particular time instant. Using the mobility profiles of the nodes generated offline from ns-2, we will be able to determine the locations of a node at any particular time instant. A static graph $G(t) = (V, E)$ generated for a particular time instant t, comprises of all the nodes in the network as the vertex set V; there exists an edge $(u, v) \in E$, if and only if, the Euclidean distance between the two end vertices u and $v \in V$, is less than or equal to the transmission range of the nodes in the network. All the edges in E are of unit weight. We assume a homogeneous network of nodes and all nodes operate at an identical and fixed transmission range. A mobile graph [7] is defined as the sequence $G_M = G_1G_2 \ldots G_T$ of static graphs that represents the network topology changes over some time scale T. In the simplest case, the mobile graph $G_M = G_1G_2 \ldots G_T$ can be extended by a new instantaneous graph

G_{T+1} to a longer sequence $G_M = G_1G_2 \ldots G_T\, G_{T+1}$, where G_{T+1} captures a link change (either a link comes up or goes down). But such an approach has very poor scalability. In this research work, we sample the network topology periodically for every 0.25 seconds, which could, in reality, be the instants of data packet origination at the source. For simplicity, we assume that all graphs in G_M have the same vertex set (i.e., no node failures).

2.2 Breadth First Search (BFS)

The BFS algorithm has been traditionally used to check the connectivity of a network graph. When we start the BFS algorithm on a randomly chosen node, we should be able to visit all the vertices in the graph, if the graph is connected. BFS returns a tree rooted at the chosen start node; when we visit a vertex v for the first time in our BFS algorithm, the vertex u through which we visit v is considered as the predecessor node of v in the tree. Every vertex in the BFS tree, other than the root node, has exactly one predecessor node. When we run BFS on a static graph with unit edge weights, we will be basically obtaining a minimum hop multicast tree such that every node in the graph is connected to the root node (the source node of the multicast group) of the tree on a path with the theoretically minimum hop count.

Figure 1 illustrates BFS in the form of a pseudo code and Figure 2 demonstrates the step-by-step execution of BFS on a sample graph. If $MG \subseteq V$ represents the multicast group – set of receiver nodes and a source node s, we start BFS at s and visit all the vertices in the network graph. Once we obtain a BFS tree rooted at s, we trace back from every receiver $d \in MG$ and determine the minimum hop s-d path. The minimum hop multicast tree is an aggregate of all these minimum hop paths connecting the source s to receiver d in the multicast group.

The set of vertices represented in parentheses below each of the graphs in Figure 2 represents the FIFO-Queue data structure used to maintain the list of vertices that are visited but whose neighbors are yet to be explored (refer the pseudo code in Figure 1). The vertices stored in this queue are extracted in a First-In First-Out fashion and their neighbors are visited if they have not been already explored. Note that, for simplicity, we restrict our research in this paper to only single source multicast groups; the research could be easily extended for multicast groups with more than one source node. Once we establish the benchmarks for single source multicast groups in this paper, we will extend the research for multi-source multicast groups in the immediate future.

```
Input: Static Graph $G = (V, E)$, source $s$
Auxiliary Variables/Initialization: Nodes-Explored = $\Phi$, FIFO-Queue = $\Phi$, root-node
                                    $\forall v \in V$, Predecessor(v) = NULL

Begin Algorithm BFS ($G$, $s$)
  root-node = $s$
  Nodes-Explored = Nodes-Explored U {root-node}
  FIFO-Queue = FIFO-Queue U {root-node}
  while ( |FIFO-Queue| > 0 ) do
    first-node $u$ = Dequeue(FIFO-Queue) // extract the first node
```

```
        for (every edge $(u, v) \in E$) do // i.e. every neighbor $v$ of node $u$
            if ($v \notin$ Nodes-Explored) then
                Nodes-Explored = Nodes-Explored U {$v$}
                FIFO-Queue = FIFO-Queue U {$v$}
                Predecessor ($v$) = $u$
            end if
        end for
    end while

End Algorithm BFS
```

Figure 1: Pseudo Code for Breadth First Search (BFS)

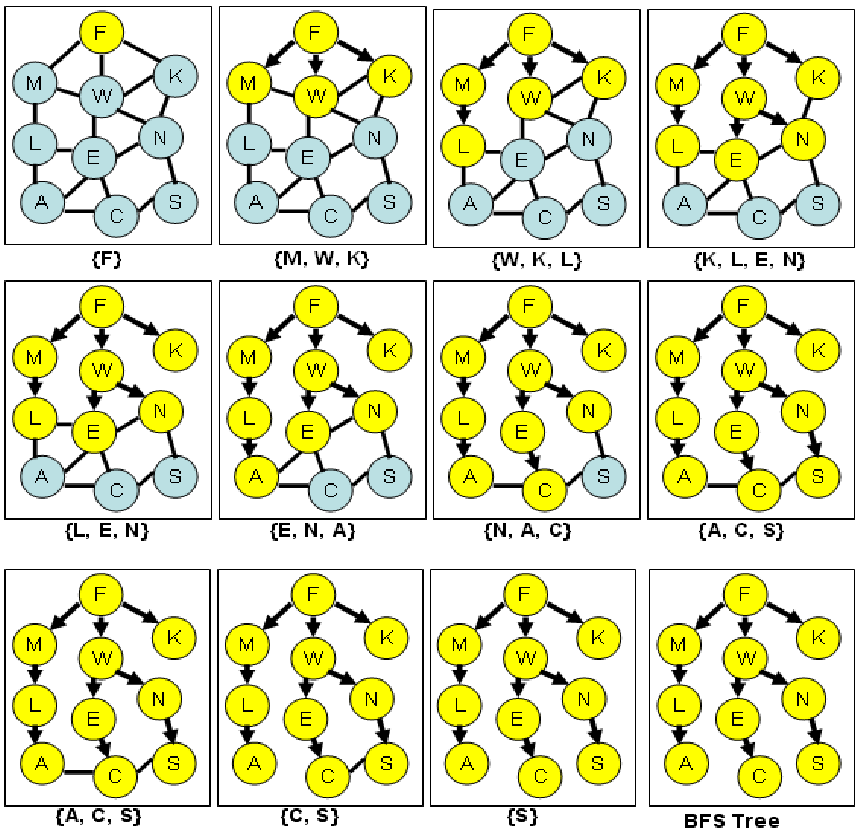

Figure 2: Execution of BFS on a Sample Graph

2.3 Minimum Edge Multicast Steiner Tree

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., [4]) have been proposed in the literature to approximate a minimum Steiner tree. In this paper, we use the Kou et al's [4] 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 $MG \subseteq V$.

We give a brief outline of the heuristic in Figure 3 and illustrate the working of the heuristic through an example in Figure 4. 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.

```
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 3: Kou et al's Heuristic [4] to find an Approximate Minimum Edge Steiner Tree

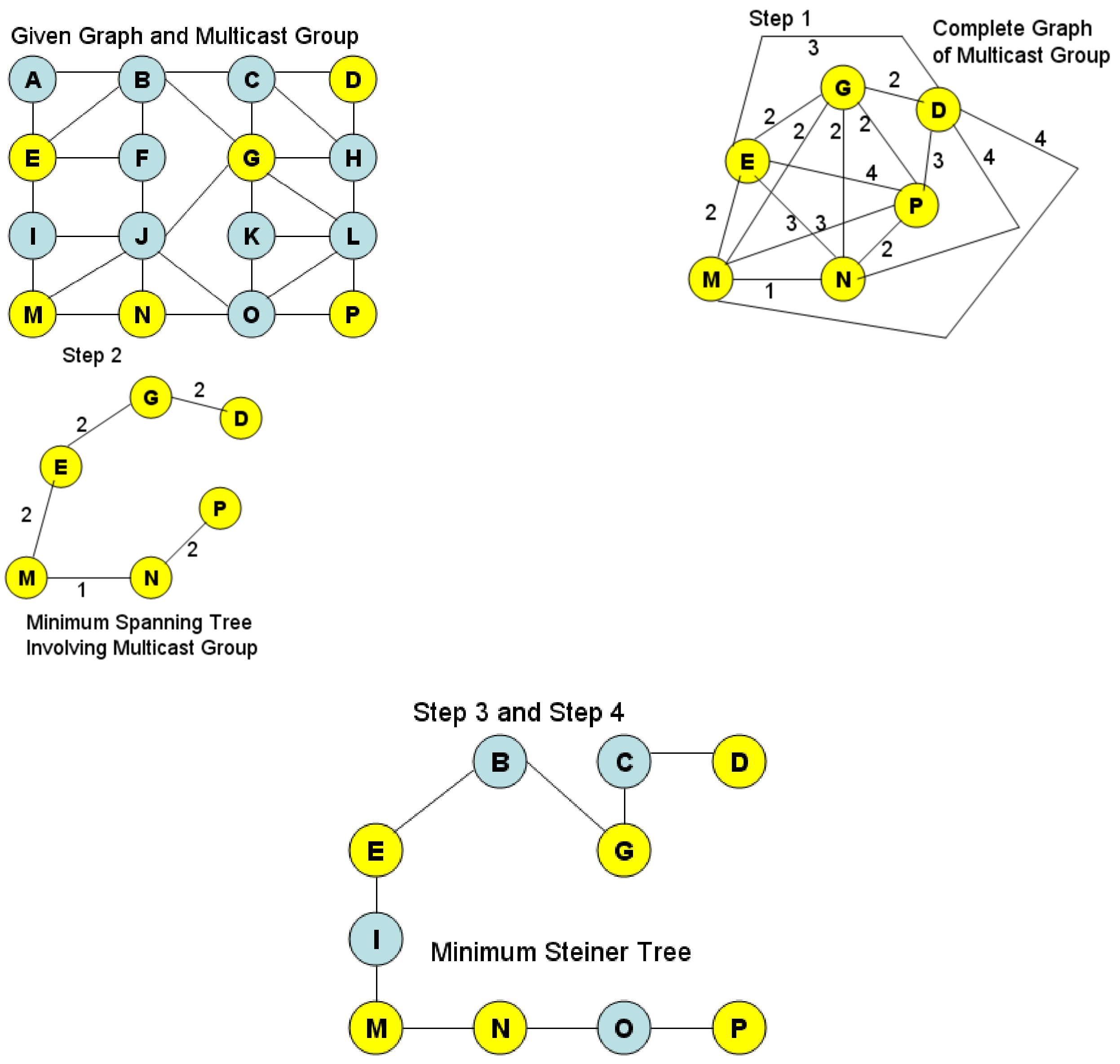
Given Graph and Multicast Group
A B C D
E F G H
I J K L
M N O P
Step 1
Complete Graph of Multicast Group
Step 2
Minimum Spanning Tree Involving Multicast Group
Step 3 and Step 4
Minimum Steiner Tree

Figure 4: Example to Illustrate the Construction of a Minimum Steiner Tree

2.4 Maximum Lifetime Multicast Tree

Given the complete knowledge of future topology changes, the algorithm *OptTreeTrans* [5], to determine a sequence of long-living multicast trees, operates on the following principle: Whenever a multicast tree connecting a given source node to all the members of a multicast group is required, choose the multicast tree that will keep the source connected to the multicast group members for the longest time. The above strategy is repeated over the duration of the multicast session and the sequence of stable multicast trees obtained by running this algorithm is called the Stable Mobile Multicast Tree. We use the BFS algorithm to determine the multicast tree in the mobile graph (static graphs representing snapshots of the network topology). We give a brief outline of the heuristic in Figure 5. An (s-MG)-tree is defined as the multicast tree connecting a source node s to all the members of the multicast group MG. Note that $s \in MG$.

Let $G_M = G_1 G_2 \ldots G_T$ be the mobile graph generated by sampling the network topology at regular instants $t_1, t_2, \ldots, t_T$ of a multicast session. When an (s-MG)-tree is required at sampling time instant t_i, the strategy is to find a mobile sub graph $G(i, j) = G_i \cap G_{i+1} \cap \ldots \cap G_j$ such that there exists at least one multicast (s-MG)-tree in $G(i, j)$ and none exists in $G(i, j+1)$. A multicast (s-MG)-tree in $G(i, j)$ is selected using the BFS algorithm. Such a tree exists in each of the static graphs G_i, G_{i+1}, …, G_j. If sampling instant $t_{j+1} \leq t_T$, the above procedure is repeated by finding the (s-MG)-tree that can survive for the maximum amount of time since t_{j+1}. A sequence of such maximum lifetime (s-MG) multicast trees over the timescale of G_M forms the Stable Mobile Multicast Tree in G_M. The pseudo code is given in Figure 5.

```
Input: $G_M = G_1 G_2 \ldots G_T$, source $s$, multicast group $MG$
Output: $(s\text{-}MG)_{StableMobileMulticastTree}$
Auxiliary Variables: $i, j$
Initialization: $i$=1; $j$=1; $(s\text{-}MG)_{StableMobileMulticastTree} = \Phi$

Begin OptTreeTrans
1 while $(i \leq T)$ do
2     Find a mobile graph $G(i, j) = G_i \cap G_{i+1} \cap \ldots \cap G_j$ such that there exists at least one ($s$-
      $MG$)-tree in $G(i, j)$ and {no ($s$-$MG$)-tree exists in $G(i, j+1)$ or $j = T$}
3     $(s\text{-}S)_{StableMobileMulticastTree} = (s\text{-}MG)_{StableMobileMulticastTree}$ U {BFS ($s$-$MG$)-tree in $G(i, j)$ }
4      $i = j + 1$
5   end while
6  return $(s\text{-}S)_{StableMobileMulticastTree}$
End OptTreeTrans
```

Figure 5: Pseudo Code for Algorithm *OptTreeTrans*

The working of the *OptTreeTrans* algorithm is illustrated using an example in Figure 6. Vertex 1 is the source node and vertices 5 and 6 are the receiver nodes of the multicast group. In the example, we can observe that there exists one stable multicast tree with edges {(1, 4), (4, 5), (5, 6)} in static graphs G_1, G_2 and G_3 and another stable multicast tree with edges {(1, 2), (2, 5), (5, 6)} in static graphs G_4 and G_5 respectively. There are three edges in both of these stable mobile multicast trees – thus the average number of edges per stable mobile multicast tree is 3.0. In both these stable mobile multicast trees, it takes two and three hops to reach respectively vertices 5 and 6 from the source vertex 1. Hence, the average hop count per source receiver path is 2.5. On the other hand, if we aim for a minimum hop multicast tree in each of the individual static graphs G_1 through G_5, we observe that the multicast tree determined in a particular static graph does not exist in the subsequent static graph and a new minimum hop multicast tree has to be determined for each of the five static graphs. There are 4, 3, 4, 4 and 3 edges in each minimum hop multicast tree and hence the average number of edges is 3.6. In each of the minimum hop multicast trees, the two vertices 5 and 6 are two hops away from the source vertex 1. Hence, the average hop count per source-receiver path is 2.0. We thus observe a tradeoff between hop count per source-receiver path and tree lifetime and this is also vindicated in our simulation analysis.

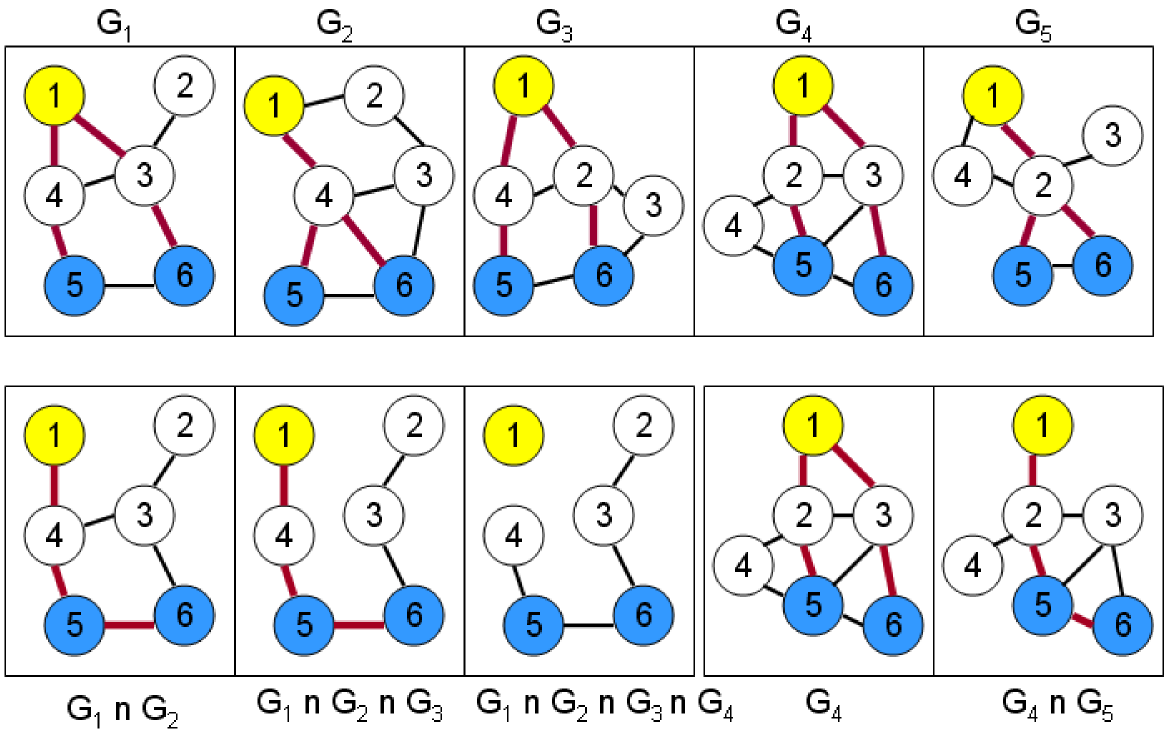

Figure 6: Example to Illustrate the Execution of Algorithm *OptTreeTrans* and the Comparison to Minimum Hop Multicast Tree

3. Simulations

The simulations have been conducted in a discrete-event simulator implemented by the author in Java. The three multicast algorithms have been implemented in a centralized fashion. We generate the static graphs by taking snapshots of the network topology, periodically for every 0.25 seconds, and run the three multicast algorithms. The simulation time is 1000 seconds. We consider a square network of dimensions 1000m x 1000m. The transmission range of the nodes is 250m. The network density is varied by performing the simulations with 50 nodes (low density) and 100 nodes (high density). We assume there is only one source for the multicast group and three different values for the number of receivers per multicast group are considered: 3 (small), 10 (moderate) and 18 (large). A multicast group comprises of a source node and a list of receiver nodes, the size of which is mentioned above.

The node mobility model used is the Random Waypoint model [8]. 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), 25 m/s (moderate mobility) and 50 m/s (high mobility). The pause time is 0 seconds.

The performance metrics measured are as follows. Each performance metric illustrated in Figures 7 through 13 is measured using 5 different lists of receiver nodes for the same size and the multicast algorithm is run on five different mobility trace files generated for a particular value of v_{max}.

(i) ***Tree Connectivity:*** This metric refers to the percentage of time instants there exists a multicast tree connecting the source node to the receiver nodes of the multicast group, averaged over the mobility profiles generated for a particular value of v_{max} for a given number of network nodes and number of receivers per multicast group.

(ii) ***Lifetime per Multicast Tree:*** Whenever a link break occurs in a multicast tree, we establish a new multicast tree. The lifetime per multicast tree is the average of the time between successive multicast tree discoveries for a particular routing protocol or algorithm, over the duration of the multicast session. The larger the value of the lifetime per multicast tree, the lower the number of multicast tree transitions or discoveries needed.

(iii) ***Number of edges per tree:*** This metric refers to the total number of edges in the entire multicast tree, time-averaged over the duration of the multicast session. For example, a multicast session uses two trees, one tree with 10 edges for 3 seconds and another tree with 15 edges for 6 seconds, then the time-averaged value for the number of edges per tree for the 9-second duration of the multicast session is (10*3 + 15*6)/(3 + 6) = 13.3 and not 12.5.

(iv) ***Number of hops per receiver:*** We measure the number of hops in the paths from the source to each receiver of the multicast group and average it for the duration of the multicast session. This metric is also a time-averaged value of the number of hops from a multicast source to a receiver and then averaged over all the receivers of a multicast session.

3.1 Tree Connectivity

The connectivity of the trees (refer Figure 7) does not depend on any individual multicast algorithm used and depends only on the network density, node mobility and the number of receivers per multicast group. For a fixed density and node mobility, as we increase the number of receivers per multicast group, the number of time instants for which we could connect the source node to all the receiver nodes decreases. With node mobility, the source may not be connected all the time to all the receivers. The probability of the source connected to all the receiver nodes decreases with increase in the number of receivers per multicast group. On the other hand, for a fixed node mobility and number of receivers per multicast group, the connectivity of a multicast tree increases with increase in the network density. This could be attributed to the availability of a larger number of nodes to connect the source node to the multicast receivers. For low density networks, we observe that as the number of receivers per multicast group increases, the percentage of tree connectivity decreases with increase in maximum node velocity. This can be attributed to an appreciable probability (in low density networks) of not being able to find a path that connects a source node to all the receiver nodes of the multicast group. As the network density increases, we do not observe relatively less variations in tree connectivity with respect to increase in the number of receivers per multicast group as well as with increase in maximum node velocity.

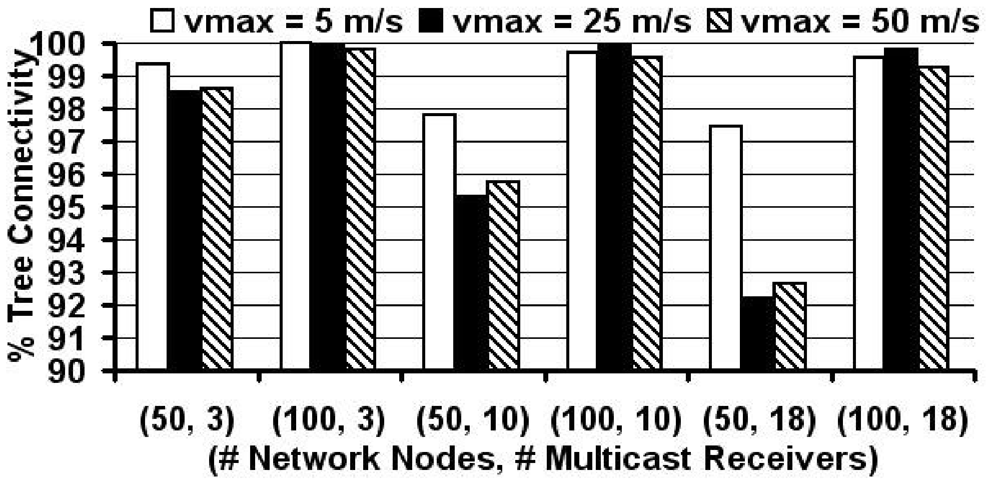

Figure 7: Tree Connectivity vs. Network Density, Node Mobility and Multicast Group Size

3.2 Lifetime per Multicast Tree

As expected, algorithm *OptTreeTrans* returned multicast trees with the theoretically maximum lifetime. However, the lifetime per stable mobile multicast tree is significantly larger compared to the lifetime per minimum hop multicast tree and minimum edge multicast tree. Hence, in Figure 8, we plotted the lifetimes of these two trees separately from the maximum possible lifetime per multicast tree. The minimum edge multicast trees had a relatively longer lifetime compared to the minimum hop multicast trees. This could be attributed to (i) the increased number of edges (refer to Section 3.3 for more on this observation) in a minimum hop multicast tree; (ii) the physical Euclidean distance between the constituent nodes of an edge on a minimum hop path is close to the transmission range of the nodes at the time of tree formation itself. Thus, the probability of an edge failure is quite high at the time of formation of the tree; (iii) the edges of a tree are also independent from each other. All these three factors play a significant role in the relatively lower lifetime per minimum hop multicast tree. For all the three multicast algorithms, for a fixed network density, as the number of receivers per multicast group is increased, the lifetime per multicast tree decreases moderately at low node mobility and decreases drastically (as large as one-half to one-third of the value at smaller number of receivers per group) at moderate and high node mobility scenarios. This could be attributed to the difficulty in finding a tree that would keep the source node connected to the receivers of the multicast group for a longer time, with increase in node mobility and/or the number of receivers per multicast group. For a given number of receivers per multicast group and node mobility, the lifetime per minimum hop trees and minimum edge trees slightly decreases as we double the network density. The decrease is more predominant for minimum hop trees and this could be attributed to the relatively unstable minimum hop paths in high density networks (refer Section 3.3 for more discussion on this observation). On the other hand, the lifetime per stable mobile multicast tree increases significantly with increase in the network density. This could be attributed to the characteristic of algorithm *OptTreeTrans* to look into the future and consider edges that will exist for a longer time. As the number of nodes in the network increases, algorithm *OptTreeTrans* has good chances of finding edges that will exist for a longer time and help to connect the source node with the receiver nodes of the multicast group.

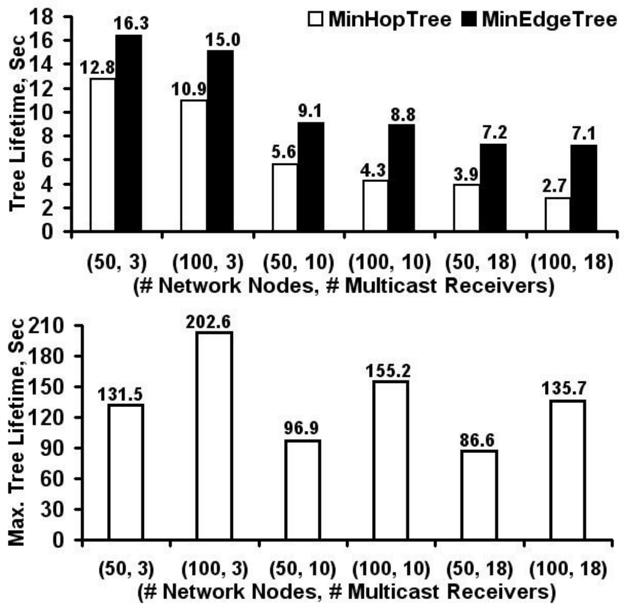

Figure 8: Lifetime of Min. Hop, Min. Edge Trees vs. Max. Tree Lifetime (v_{max} = 5 m/s)

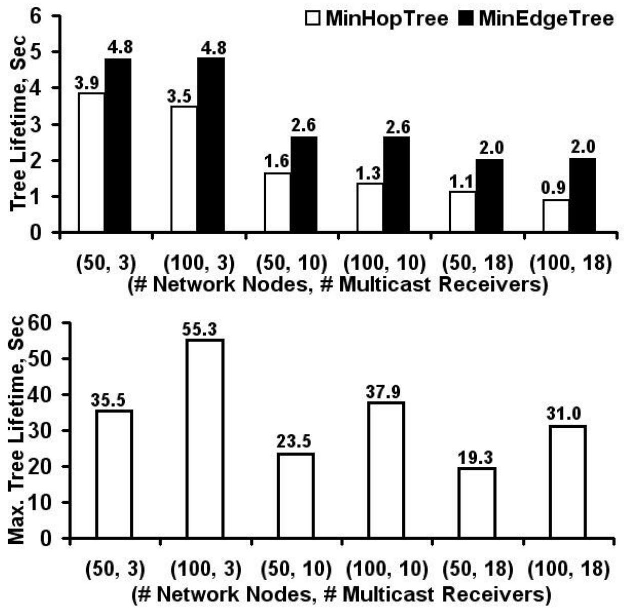

Figure 9: Lifetime of Min. Hop, Min. Edge Trees vs. Max. Tree Lifetime (v_{max} = 25 m/s)

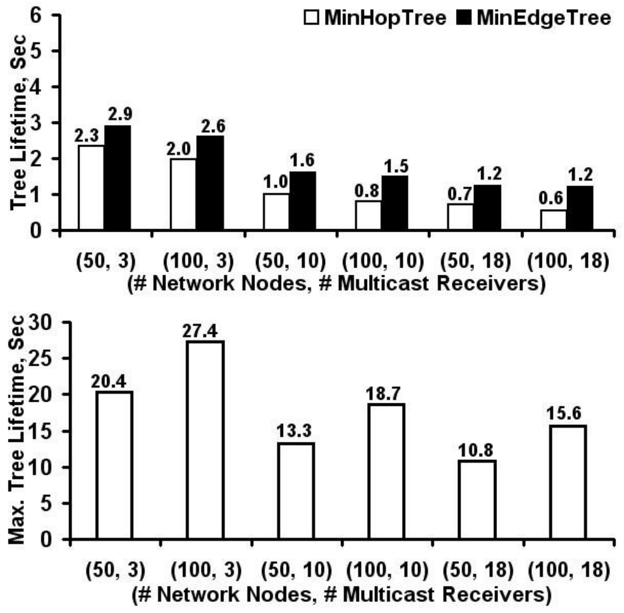

Figure 10: Lifetime of Min. Hop, Min. Edge Trees vs. Max. Tree Lifetime (v_{max} = 50 m/s)

For a given level of node mobility, the lifetime per minimum edge tree is 23% (low density) to 38% (high density); 61% (low density) to 107% (high density) and 76% (low density) to 160% (high density) larger than the lifetime per minimum hop tree for small, moderate and larger number of receivers per multicast group respectively. However, compared to the maximum tree lifetime returned by algorithm *OptTreeTrans*, the lifetime per minimum edge multicast tree is only 5% (higher density, larger number of receivers per multicast group) to 14% (lower density, smaller number of receivers per multicast group) of the maximum possible multicast tree lifetime. Thus, the minimum hop and minimum edge trees have significantly lower lifetime compared to the maximum possible lifetime per multicast tree. For both minimum hop and minimum edge trees, for a given network density and number of receivers per group, as we increase the maximum node velocity to 25 m/s and 50 m/s, the lifetime per tree reduces by 1/3rd to 1/6th of their value at a maximum node velocity of 5 m/s. On the other hand, the lifetime of the stable multicast trees reduces by 1/4th to 1/8th.

3.3 Number of Edges per Multicast Tree and Hop Count per Source-Receiver Path

As expected, the minimum-edge based Steiner trees incurred the smallest number of edges per multicast trees. The maximum lifetime multicast trees of algorithm *OptTreeTrans* incurred more edges compared to the minimum hop based multicast trees. The number of edges per minimum hop tree and maximum lifetime tree is, on average, 13-35% and 35-68% larger than those incurred with the minimum edge tree. Even though, the maximum lifetime multicast trees have a larger number of edges than the minimum hop trees, the former are relatively more stable than the latter. This could be due to the look-ahead approach of algorithm *OptTreeTrans* to look into the future and select edges that have a longer lifetime. On the other hand, with an objective to optimize the hop count, minimum hop based multicast trees select edges that could constitute a minimum hop path, but with a higher probability of failure in the immediate future. The physical Euclidean distance between the constituent nodes of an edge on a minimum hop path is close to the transmission range of the nodes at the time of tree formation itself.

For a given network density, as we increase the number of receivers per multicast group from 3 to 18, the average number of edges per multicast tree increased by a factor of 3 to 4. For the minimum hop and minimum edge trees, for a given level of node mobility and number of receivers per multicast group, as we increase the network density, the number of edges per multicast tree either remains the same or even slightly decreases. On the other hand, in the case of the maximum lifetime multicast trees, as we increase the network density for a fixed number of receivers per multicast group, the number of edges per multicast tree increases by 10-15% and this contributes to a significantly larger lifetime. Algorithm *OptTreeTrans* cleverly makes use of the availability of a larger number of nodes to find edges that are relatively more stable.

As expected, the minimum hop multicast trees incurred the lowest hop count per source-receiver path. For smaller number of receivers per group, the hop count per source-receiver path in the maximum lifetime multicast trees was the largest. However, for moderate and larger number of receivers per multicast group, the hop count per source-receiver path in the minimum edge multicast tree was the largest. This could be attributed to a relatively lower number of edges per minimum edge multicast tree compared to the other two trees. As we connect the source node to the multicast receivers with the lowest possible number of edges, the number of hops between the source node and to each of the receiver nodes increases. This is the tradeoff between the objectives of minimizing the number of edges per multicast tree and the hop count per individual source-receiver paths in the multicast tree. On the other hand, the maximum lifetime multicast trees incur an increase in the number of edges with increase in the network density and/or the number of receivers per multicast group. As a result, it would be then possible to connect the source node to each of the receiver nodes with a relatively lower hop count path, compared to those incurred when the number of receivers per multicast group is smaller.

For both minimum hop and minimum edge multicast trees, for a given network density and number of receivers per multicast group, there is appreciably no impact of the maximum node velocity on the average number of edges per tree as well as the hop count per source-receiver path. On the other hand, for the maximum lifetime multicast trees, for a given network density and number of receivers per multicast group, as we increase the maximum node velocity, the average number of edges per tree decreases by at most 14% and the hop count per source-receiver path decreases by 10-18%. This could be attributed to the dynamically changing network topology at higher node velocities and there is no need for algorithm *OptTreeTrans* to look ahead into the far future and include more robust edges and multi-hop paths to find long-living trees. It would not be possible to find long-living stable edges at higher node velocities.

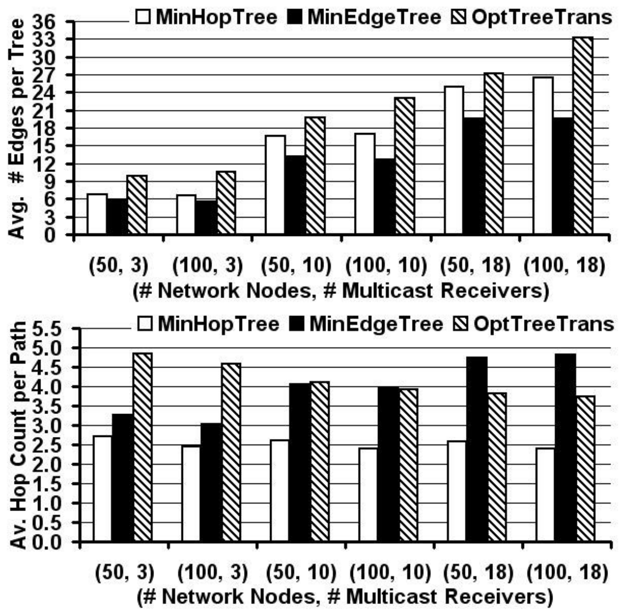

Figure 11: # Edges per Multicast Tree and Hop Count per Source-Receiver Path (v_{max} = 5 m/s)

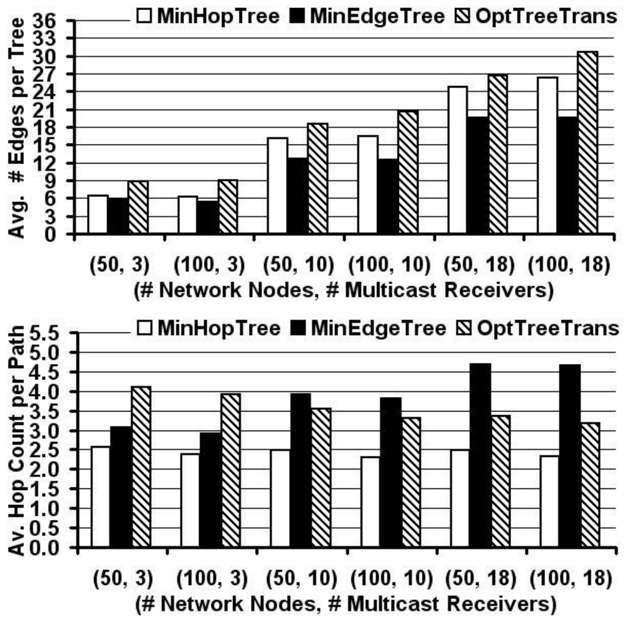

Figure 12: # Edges per Multicast Tree and Hop Count per Source-Receiver Path (v_{max} = 25 m/s)

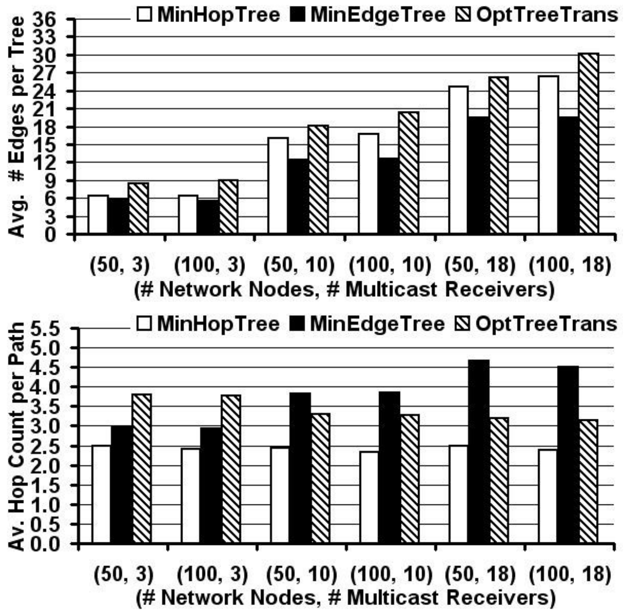

Figure 13: # Edges per Multicast Tree and Hop Count per Source-Receiver Path (v_{max} = 50 m/s)

For a given level of node mobility (i.e., maximum node velocity) and network density, as we increase the number of receivers per multicast group, the average hop count per source-receiver path for minimum hop trees and maximum lifetime trees decreases. On the other hand, the average hop count per source-receiver path for minimum edge trees increases. This could be attributed to the relatively fewer number of edges in the minimum edge trees compared to those incurred by the minimum hop and maximum lifetime trees. The relatively larger number of edges in the minimum hop and maximum lifetime trees at larger number of receivers per multicast group results in lower hops count per source-receiver path. The average number of edges per minimum hop tree and maximum lifetime tree for a network of 50 nodes and 3 receivers per multicast group is respectively about 1 and 4 edges more than those incurred by the minimum edge trees; on the other hand, the average number of edges per minimum hop tree and maximum lifetime tree for a network of 50 nodes and 18 receivers per multicast group is respectively about 7 and 14 edges more than the minimum. Similar observations could be made for network of 100 nodes.

When compared to the average hop count per source-receiver path incurred by minimum hop trees, the average hop count per source-receiver path for minimum edge trees is 20% (for smaller number of receivers per multicast group) to 100% (for larger number of receivers per multicast group) more. On the other hand, the average hop count per source-receiver path for maximum lifetime trees is 28% (for larger number of receiver per multicast group) to 86% (for smaller number of receivers per multicast group) more than the average minimum hop count per path. Note that with increase in the network density and/or the number of receivers per multicast group, the trend of the hop counts per source-receiver path for both minimum hop and maximum lifetime trees is to decrease; whereas, the trend of the hop count per source-receiver path for minimum edge trees is to increase. The hop count per source-receiver path for minimum hop and maximum lifetime trees decreases by at most 14% and 30% respectively; whereas, the hop count per source-receiver path for minimum edge trees increases by at most 47%.

4. Conclusions

We have described the algorithms that can be used to obtain benchmarks for the minimum hop count per source-receiver path, minimum number of edges per tree and maximum lifetime per tree for multicast routing in mobile ad hoc networks. Simulations have been conducted to obtain such benchmarks for different conditions of network density, node mobility and number of receivers per multicast group. The minimum hop and maximum lifetime based multicast trees have a larger number of edges than the theoretical minimum – the minimum hop trees are unstable and their lifetime decreases with increase in the number of edges; the duration of existence of the maximum lifetime stable multicast trees increases with increase in the number of edges. The former could be attributed to the instantaneous decision taken by the minimum hop path algorithm to select a tree without any consideration for the number of edges and their lifetime; while, the latter could be correlated to the look-ahead approach of algorithm *OptTreeTrans* to find a tree involving long-living stable edges. The minimum edge trees have a relatively larger hop count per source-receiver path and the hop count per path increases with the number of receivers per multicast group. The relatively fewer edges in the minimum edge tree results in a relatively larger lifetime compared to the minimum hop trees, as each edge in these two trees are independent. Nevertheless, the lifetime of both the minimum edge and minimum hop trees is significantly smaller than the theoretically maximum lifetime per multicast tree, as discovered using algorithm *OptTreeTrans*.

The simulation results thus indicate a complex tradeoff between the hop count per source-receiver paths, number of edges and lifetime per tree for multicast routing. It is not possible to optimize all the three metrics with a single multicast algorithm and a multicast algorithm optimized for one particular metric returns sub-optimal performance with respect to the other two metrics. This motivates us to explore routing algorithms and protocols that could minimize the tradeoff between two or all the three metrics rather than just optimizing one of them, at the cost of others.

References

[1] C. Siva Ram Murthy and B. S. Manoj, "Routing Protocols for Ad Hoc Wireless Networks," *Ad Hoc Wireless Networks: Architectures and Protocols*, Chapter 7, pp. 299 – 364, Prentice Hall, 1st Edition, June 2004.

[2] C. K. Toh, G. Guichal and S. Bunchua, "ABAM: On-demand Associatvity-based Multicast Routing for Ad hoc Mobile Networks," *Proceedings of the 52nd IEEE VTS Fall Vehicular Technology Conference*, Vol. 3, pp. 987 – 993, September 2000.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, "Introduction to Algorithms," 2nd Edition, MIT Press/ McGraw-Hill, Sept. 2001.

[4] L. Kou, G. Markowsky and L. Berman, "A Fast Algorithm for Steiner Trees," Vol 15, pp. 141-145, *Acta Informatica*, Springer-Verlag, 1981.

[5] N. Meghanathan, "On the Stability of Paths, Steiner Trees and Connected Dominating Sets in Mobile Ad hoc Networks," *Ad Hoc Networks*, Vol. 6, No. 5, pp. 744-769, July 2008.

[6] K. Fall, K. Varadhan, "The ns Manual," The VINT Project, A Collaboration between researchers at UC Berkeley, LBL, USC/ISI and Xerox PARC.

[7] A. Farago and V. R. Syrotiuk, "MERIT: A Scalable Approach for Protocol Assessment," *Mobile Networks and Applications*, Vol. 8, No. 5, pp. 567 – 577, October 2003.

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