14

Greedy Algorithms to Determine Stable Paths and Trees in Mobile Ad hoc Networks

Natarajan Meghanathan
Jackson State University, Jackson, MS
United States of America

1. Introduction

A mobile ad hoc network (MANET) is a dynamic, resource-constrained, distributed system of independent and arbitrarily moving wireless nodes and bandwidth-constrained links. MANET nodes operate with limited battery charge and use a limited transmission range to sustain an extended lifetime. As a result, MANET routes are often multi-hop in nature and nodes forward the data for others. Based on their primary route selection principle, MANET routing protocols are classified into two categories (Meghanathan & Farago, 2004): minimum-weight based routing and stability-based routing protocols. The minimum-weight path among the set of available paths in a weighted network graph is the path with the minimum total weight summed over all its edges. The routing metrics generally targeted include: hop count, delay, energy consumption, node lifetime and etc. The stability-based routing protocols are aimed to minimize the number of route transitions and incur the least possible route discovery and maintenance overhead to the network.

A majority of the ad hoc routing protocols are minimum-weight based and are proposed to optimize one or more performance metrics in a greedy fashion without looking at the future. For example, the Dynamic Source Routing (DSR) protocol (Johnson et. al., 2001) instantaneously selects any shortest path that appears to exist and similarly the Ad hoc On-demand Distance Vector (AODV) protocol (Perkins & Royer, 1999) chooses the route that propagated the Route Request (RREQ) route discovery messages, with the lowest delay. To maintain optimality in their performance metrics, minimum-weight based routing protocols change their paths frequently and incur a huge network overhead. The stability-based routing protocols attempt to discover stable routes based on the knowledge of the past topology changes, future topology changes or a combination of both. Prominent within the relatively smaller class of stability-based routing protocols proposed in the literature include: Associativity-based Routing (ABR) (Toh, 1997), Flow-Oriented Routing Protocol (FORP) (Su et. al., 2001) and the Route-lifetime Assessment Based Routing (RABR) (Agarwal et. al., 2000) protocols. ABR selects paths based on the degree of association stability, which is basically a measure of the number of beacons exchanged between two neighbor nodes. FORP selects the route that will have the largest expiration time since the time of its discovery. The expiration time of a route is measured as the minimum of the predicted expiration time of its constituent links. RABR uses the average change in the received signal strength to predict the time when the received signal strength would fall below a critical

Source: Advances in Greedy Algorithms, Book edited by: Witold Bednorz,
ISBN 978-953-7619-27-5, pp. 586, November 2008, I-Tech, Vienna, Austria

threshold. The stable path MANET routing protocols are distributed and on-demand in nature and thus are not guaranteed to determine the most stable routes (Meghanathan 2006d; Meghanathan 2007).

Stability is an important design criterion to be considered while developing multi-hop MANET routing protocols. The commonly used route discovery approach of flooding the route request can easily lead to congestion and also consume node battery power. Frequent route changes can also result in out-of-order data packet delivery, causing high jitter in multi-media, real-time applications. In the case of reliable data transfer applications, failure to receive an acknowledgement packet within a particular timeout interval can also trigger retransmissions at the source side. As a result, the application layer at the receiver side might be overloaded in handling out-of-order, lost and duplicate packets, leading to reduced throughput. Thus, stability is also important from quality of service (QoS) point of view too.

This chapter addresses the issue of finding the sequence of stable paths and trees, such that the number of path and tree transitions is the global minimum. In the first half of the chapter, we present an algorithm called *OptPathTrans* (Meghanathan & Farago, 2005) to determine the sequence of stable paths for a source-destination (*s-d*) communication session. Given the complete knowledge of the future topology changes, the algorithm operates on the greedy "look-ahead" principle: Whenever an *s-d* path is required at a time instant *t*, choose the longest-living *s-d* path from *t*. The sequence of long-living stable paths obtained by applying the above strategy for the duration of the *s-d* session is called the stable mobile path and it incurs the minimum number of route transitions. We quantify route stability in terms of the number of route transitions. Lower the number of route transitions, higher is the stability of the routing algorithm.

In the second half of the chapter, we show that the greedy look-ahead principle behind *OptPathTrans* is very general and can be extended to find a stable sequence of any communication structure as long as there is an underlying algorithm or heuristic to determine that particular communication structure. In this direction, we propose algorithm *OptTreeTrans* (Meghanathan, 2006c) to determine the sequence of stable multicast Steiner trees for a multicast session. The problem of determining the multicast Steiner tree is that given a weighted network graph $G = (V, E)$ where V is the set of vertices, E is the set of edges connecting these vertices and S, is a subset of set of vertices V, called the multicast group or Steiner points, we want to determine the set of edges of G that can connect all the vertices of S and they form a tree. It is very rare that greedy strategies give an optimal solution. Algorithms *OptPathTrans* and *OptTreeTrans* join the league of Dijkstra algorithm, Minimum spanning tree Kruskal and Prim algorithms (Cormen et. al., 2001) that have used greedy strategies, but yet give optimal solution. In another related work, we have also proposed an algorithm to determine the sequence of stable connected dominating sets for a network session (Meghanathan, 2006b).

The performance of algorithms *OptPathTrans* and *OptTreeTrans* have been studied using extensive simulations under two different scenarios: (1) Scenarios in which the complete knowledge of the future topology changes is available at the time of path/tree selection and (2) Scenarios in which the locations of nodes are only predicted for the near future and not exact. To simulate the second scenario, we consider a location prediction model called "Prediction with Uncertainty" that predicts the future locations of nodes at different time instants based on the current location, velocity and direction of travel of each node, even though we are not certain of the velocity and direction of travel in the future. Simulation

results illustrate that the algorithms *OptPathTrans* and *OptTreeTrans*, when run under the limited knowledge of future topology changes, yield the sequence of paths and trees such that the number of transitions is close to the minimum values obtained when run under the complete knowledge of future topology changes.

The rest of the chapter is organized as follows: In Section 2, we describe algorithm *OptPathTrans* to determine the stable mobile path, discuss its proof of correctness and run-time complexity. Section 3 illustrates the simulation results of *OptPathTrans* under the two scenarios of complete and limited knowledge of future topology changes. In Section 4, we explain algorithm *OptTreeTrans* to determine the stable mobile multicast Steiner tree, discuss its proof of correctness and run-time complexity. Section 5 illustrates the simulation results of *OptTreeTrans* under the two scenarios of complete and limited knowledge of future topology changes. In Section 6, we discuss the impact of the stability-hop count tradeoff on network resources and routing protocol performance. Section 7 concludes the chapter and discusses future work. Note that we use the terms 'path' and 'route' interchangeably throughout the chapter. They are the same.

2. Algorithm for the optimal number of path transitions

One could resort to flooding as a viable alternative at high mobility (Corson & Ephremides, 1995). But, flooding of the data packets will prohibitively increase the energy consumption and congestion at the nodes. This motivates the need for stable path routing algorithms and protocols in dynamically changing scenarios, typical to that of MANETs.

2.1 Mobile graph

A mobile graph (Farago & Syrotiuk, 2003) 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 chapter, we sample the network topology periodically for every one second, 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 Mobile path

A *mobile path* (Farago & Syrotiuk, 2003), defined for a source-destination (*s*-*d*) pair, in a mobile graph $G_M = G_1G_2 \ldots G_T$ is the sequence of paths $P_M = P_1P_2 \ldots P_T$, where P_i is a static path between the same *s*-*d* pair in $G_i = (V_i, E_i)$, V_i is the set of vertices and E_i is the set of edges connecting these vertices at time instant t_i. That is, each static path P_i can be represented as the sequence of vertices $v_0v_1 \ldots v_l$, such that $v_0 = s$ and $v_l = d$ and $(v_{j-1},v_j) \in E_i$ for $j = 1,2, \ldots, l$. The timescale of t_T normally corresponds to the duration of an *s*-*d* session.

Let $w_i(P_i)$ denote the weight of a static path P_i in G_i. For additive path metrics, such as hop count and end-to-end delay, $w_i(P_i)$ is simply the sum of the link weights along the path. Thus, for a given *s*-*d* pair, if $P_i = v_0v_1 \ldots v_l$ such that $v_0 = s$ and $v_l = d$,

$$w_i(P_i) = \sum_{j=1}^{l} w_i(v_{j-1}, v_j) \tag{1}$$

For a given mobile graph $G_M = G_1G_2 \ldots G_T$ and *s-d* pair, the weight of a mobile path $P_M = P_1P_2 \ldots P_T$ is

$$w(P_M) = \sum_{i=1}^{T} w_i(P_i) + \sum_{i=1}^{T-1} C_{trans}(P_i, P_{i+1}) \tag{2}$$

where $C_{trans}(P_i, P_{i+1})$ is the transition cost incurred to change from path P_i in G_i to path P_{i+1} in G_{i+1} and is measured in the same unit used to compute $w_i(P_i)$.

2.3 Stable mobile path and minimum hop mobile path

The Stable Mobile Path for a given mobile graph and *s-d* pair is the sequence of static *s-d* paths such that the number of route transitions is as minimum as possible. A Minimum Hop Mobile Path for a given mobile graph and *s-d* pair is the sequence of minimum hop static *s-d* paths. With respect to equation (2), a Stable Mobile Path minimizes only the sum of the transition costs $\sum_{i=1}^{T-1} C_{trans}(P_i, P_{i+1})$ and a Minimum Hop Mobile Path minimizes only the term $\sum_{i=1}^{T} w_i(P_i)$, assuming unit edge weights. For additive path metrics and a constant transition cost, a dynamic programming approach to optimize the weight of a mobile path $w(P_M) = \sum_{i=1}^{T} w_i(P_i) + \sum_{i=1}^{T-1} C_{trans}(P_i, P_{i+1})$ has been proposed in (Farago & Syrotiuk, 2003).

2.4 Algorithm description

Algorithm *OptPathTrans* operates on the following greedy strategy: Whenever a path is required, select a path that will exist for the longest time. Let $G_M = G_1G_2 \ldots G_T$ be the mobile graph generated by sampling the network topology at regular instants $t_1, t_2, \ldots, t_T$ of an *s-d* session. When an *s-d* path 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 *s-d* path in $G(i, j)$ and no *s-d* path exists in $G(i, j+1)$. A minimum hop *s-d* path in $G(i, j)$ is selected. Such a path exists in each of the static graphs $G_i, G_{i+1}, \ldots, G_j$. If sampling instant $t_{j+1} \leq t_T$, the above procedure is repeated by finding the *s-d* path that can survive for the maximum amount of time since t_{j+1}. A sequence of such maximum lifetime static *s-d* paths over the timescale of a mobile graph G_M forms the Stabile Mobile *s-d* Path in G_M. The pseudo code of the algorithm is given in Fig. 1.

Input: $G_M = G_1G_2 \ldots G_T$, source *s*, destination *d*
Output: P_S // Stable Mobile Path
Auxiliary Variables: *i*, *j*
Initialization: $i=1$; $j=1$; $P_S = \Phi$

Begin *OptPathTrans*

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-d* path in $G(i, j)$ and {no *s-d* path exists in $G(i, j+1)$ or $j = T$}

3 $P_S = P_S \cup \{$ minimum hop *s-d* path in $G(i, j)\}$

4 $i = j + 1$

5 **end while**

6 **return** P_S

End *OptPathTrans*

Fig. 1. Pseudo code for algorithm *OptPathTrans*

2.5 Algorithm complexity and proof of correctness

In a mobile graph $G_M = G_1G_2 \ldots G_T$, the number of route transitions can be at most T. A path-finding algorithm will have to be run T times, each time on a graph of n nodes. If we use Dijkstra algorithm that has a worst-case run-time complexity of $O(n^2)$, where n is the number of nodes in the network, the worst-case run-time complexity of *OptPathTrans* is $O(n^2T)$. We use the proof by contradiction technique to prove the correctness of algorithm *OptPathTrans*. Let P_S (with m route transitions) be the mobile path generated by algorithm *OptPathTrans*. To prove m is optimal, we assume the contrary that there exists a mobile path $P_{S'}$ with m' route transitions such that $m' < m$. Let $epoch_S^1$, $epoch_S^2$, …., $epoch_S^m$ be the set of sampling time instants in each of which the mobile path P_S suffers no route transitions (refer Fig. 2). Similarly, let $epoch_{S'}^1$, $epoch_{S'}^2$, …, $epoch_{S'}^{m'}$ be the set of sampling time instants in each of which the mobile path $P_{S'}$ suffers no route transitions (refer Fig. 3).

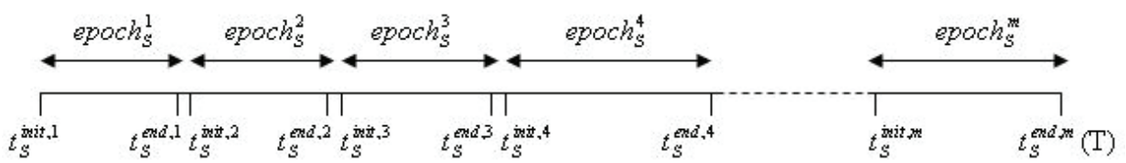


Fig. 2. Sampling Time Instants for Mobile Path P_S (Determined by *OptPathTrans*)

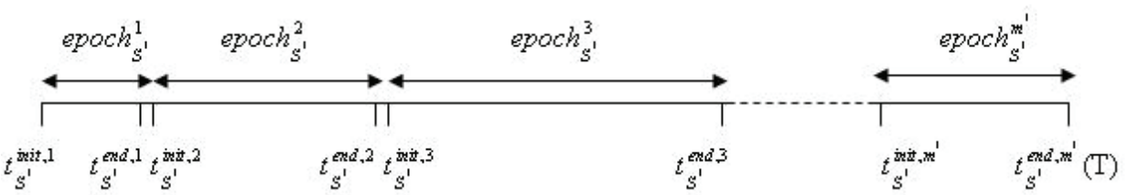


Fig. 3. Sampling Time Instants for Mobile Path $P_{S'}$ (Hypothesis for the Proof)

Let $t_S^{init,j}$ and $t_S^{end,j}$ be the initial and final sampling time instants of $epoch_s^j$ where $1 \le j \le m$. Similarly, let $t_{S'}^{init,k}$ and $t_{S'}^{end,k}$ be the initial and final sampling time instants of $epoch_{s'}^k$, where $1 \le k \le m'$. Note that $t_S^{init,1} = t_{S'}^{init,1}$ and $t_S^{end,m} = t_{S'}^{end,m'}$ to indicate that P_S and $P_{S'}$ span over the same time period, T, of the network session.

Now, since the hypothesis is $m' < m$, there should exist j, k where $1 \le j \le m$ and $1 \le k \le m'$ such that $epoch_S^j$ s $epoch_{S'}^k$, i.e., $t_{S'}^{init,k} < t_S^{init,j} < t_S^{end,j} < t_{S'}^{end,k}$ and at least one s-d path existed in $[t_{S'}^{init,k}, \ldots, t_{S'}^{end,k}]$. In other words, there should be at least one s-d path in $P_{S'}$ that has a lifetime larger than that of the lifetime of the s-d paths in P_S. But, algorithm *OptPathTrans* made a route transition at $t_S^{end,j}$ since there was no s-d path from $t_S^{init,j}$ beyond $t_S^{end,j}$. Thus, there is no common s-d path in the range $[t_S^{init,j}, \ldots, t_{S'}^{end,k}]$ and hence there is no common s-d path in the range $[t_{S'}^{init,k}, \ldots, t_{S'}^{end,k}]$. This shows that the lifetime of each of the s-d paths in $P_{S'}$ has to be smaller or equal to the lifetime of the s-d paths in P_S, implying $m' \ge m$. This is a contradiction and proves that our hypothesis $m' < m$ is not correct. Hence, the number of route transitions in P_S is optimal and P_S is the Stable Mobile Path.

2.6 Example run of algorithm *OptPathTrans*

Consider the mobile graph $G_M = G_1G_2G_3G_4G_5$ (Fig. 4.), generated by sampling the network topology for every second. Let node 1 and node 6 be the source and destination nodes respectively. The Minimum Hop Mobile 1-6 Path for the mobile graph G_M would be {{1-3-6}G_1, {1-4-6}G_2, {1-2-6}G_3, {1-3-6}G_4, {1-2-6}G_5}. As the minimum hop path in one static graph does not exist in the other, the number of route transitions incurred for the Minimum Hop Mobile Path is 5. The hop count in each of the static paths is 2 and hence the time averaged hop count would also be 2.

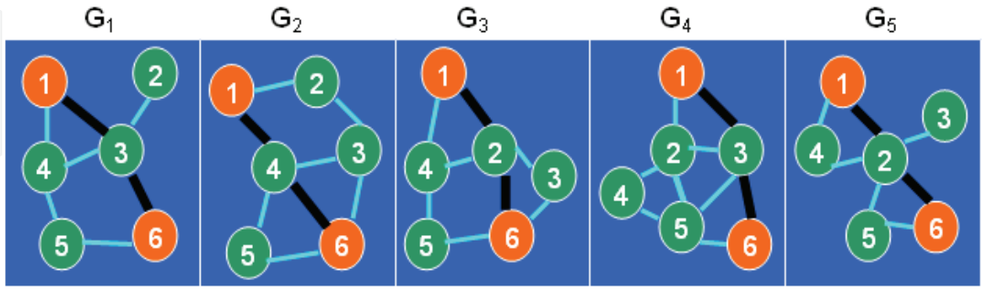


Fig. 4. Mobile Graph and Minimum Hop Mobile Path

The execution of algorithm *OptPathTrans* on the mobile graph G_M, of Fig. 4. is shown in Fig. 5. The Stable Mobile Path generated would be {{1-4-5-6}G_{123}, {1-2-5-6}G_{45}}. The number of route transitions is 2 as we have to discover a common path for static graphs G_1, G_2 and G_3 and a common path for static graphs G_4 and G_5. The hop count of each of the constituent paths of the Stable Mobile Path is 3 and hence the time averaged hop count of the Stable

Mobile Path would also be 3. Note that even though there is a 2-hop path {1-3-6} common to graphs G_1 and G_2, the algorithm ends up choosing the 3-hop path {1-4-5-6} that is common to graphs G_1, G_2 and G_3. This shows the greedy nature of algorithm *OptPathTrans*, i.e., choose the longest living path from the current time instant. To summarize, the Minimum Hop Mobile Path incurs 5 path transitions with an average hop count of 2; while the Stable Mobile Path incurs 2 path transitions with an average hop count of 3. This illustrates the tradeoff between stability and hop count which is also observed in the simulations.

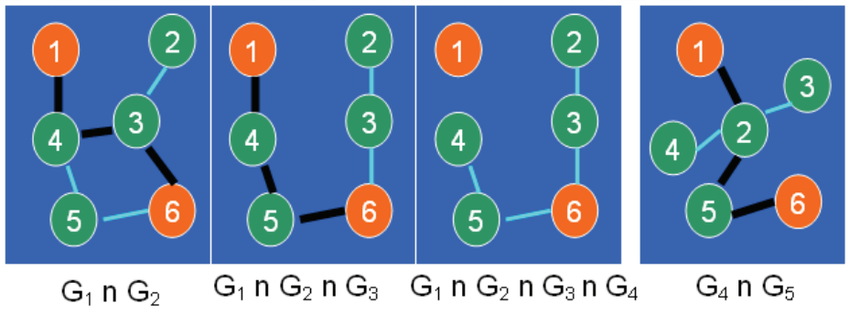


Fig. 5. Execution of Algorithm *OptPathTrans* on Mobile Graph of Fig. 4.

2.7 Prediction with uncertainty

Under the Prediction with Uncertainty model, we generate a sequence of predicted network topology changes starting from the time instant a path is required. We assume we know only the current location, direction and velocity of movement of the nodes and that a node continues to move in that direction and velocity. Whenever the node hits a network boundary, we predict it stays there, even though a node might continue to move. Thus, even though we are not sure of actual locations of the nodes in the future, we construct a sequence of predicted topology changes based on the current information. We run algorithm *OptPathTrans* on the sequence of predicted future topology changes generated starting from the time instant a path is required. We validate the generated path with respect to the actual locations of the nodes in the network. Whenever a currently used path is found to be invalid, we repeat the above procedure. The sequence of paths generated by this approach is referred to as Stable-Mobile-Path$^{\text{Uncertain-Pred}}$.

In practice, information about the current location, direction and velocity of movement could be collected as part of the Route-Request and Reply cycle in the route setup phase. After collecting the above information from each node, the source and destination nodes of a session assume that each node continues to move in its current direction of motion with the current velocity. Given the network dimensions (0… X_{max}, 0… Y_{max}), the location (x_i^t , y_i^t) of a node i at time instant t, the direction of motion Θ ($0 \le \Theta \le 360$) with reference to the positive x-axis, and the current velocity v_i^t , the location of node i at time instant $t + \delta t$, ($x_i^{t+\delta t}$, $y_i^{t+\delta t}$) would be predicted as follows:

$$x_i^{t+\delta t} = x_i^t + (v_i^t * \delta t * \cos\Theta) \quad \text{if } 0 \le \Theta \le 90$$

$$= x_i^t - (v_i^t * \delta t * \cos(180 - \Theta)) \quad \text{if } 90 \le \Theta \le 180$$

$$= x_i^t - (v_i^t * \delta t * \cos(\Theta - 180)) \quad \text{if } 180 \le \Theta \le 270$$

$$= x_i^t + (v_i^t * \delta t * \cos(360 - \Theta)) \quad \text{if } 270 \le \Theta \le 360$$

$$y_i^{t+\delta t} = y_i^t + (v_i^t * \delta t * \sin\Theta) \quad \text{if } 0 \le \Theta \le 90$$

$$= y_i^t + (v_i^t * \delta t * \sin(180 - \Theta)) \quad \text{if } 90 \le \Theta \le 180$$

$$= y_i^t - (v_i^t * \delta t * \sin(\Theta - 180)) \quad \text{if } 180 \le \Theta \le 270$$

$$= y_i^t - (v_i^t * \delta t * \sin(360 - \Theta)) \quad \text{if } 270 \le \Theta \le 360$$

At any situation, when $x_i^{t+\delta t}$ is predicted to be less than 0, then $x_i^{t+\delta t}$ is set to 0.

when $x_i^{t+\delta t}$ is predicted to be greater than X_{max}, then $x_i^{t+\delta t}$ is set to X_{max}.

Similarly, when $y_i^{t+\delta t}$ is predicted to be less than 0, $y_i^{t+\delta t}$ is set to 0.

when $y_i^{t+\delta t}$ is predicted to be greater than Y_{max}, then $y_i^{t+\delta t}$ is set to Y_{max}.

When a source-destination (*s-d*) path is required at time instant *t*, we try to find the minimum hop *s-d* path in the predicted mobile sub graph $G^{pred}(t, t+\delta t) = G_t \cap G_{t+1}^{pred} \cap G_{t+2}^{pred} \cap \ldots \cap G_{t+\delta t}^{pred}$. If a minimum hop *s-d* path exists in $G^{pred}(t, t+\delta t)$, then that path is validated in the actual mobile sub graph $G^{actual}(t, t+\delta t) = G_t \cap G_{t+1} \cap G_{t+2} \cap \ldots \cap G_{t+\delta t}$ that spans time instants *t*, *t*+1, *t*+2, …, *t*+δt. If an *s-d* path exists in both $G^{pred}(t, t+\delta t)$ and $G^{actual}(t, t+\delta t)$, then that *s-d* path is used at time instants *t*, *t*+1, …, *t*+δt.

If an *s-d* path exists in $G^{pred}(t, t+\delta t)$, $G^{pred}(t, t+\delta t+1)$ and $G^{actual}(t, t+\delta t)$, but not in $G^{actual}(t, t+\delta t+1)$, the above procedure is repeated by predicting the locations of nodes starting from time instant $t+\delta t+1$. Similarly, if an *s-d* path exists in $G^{pred}(t, t+\delta t)$ and $G^{actual}(t, t+\delta t)$, but not in $G^{pred}(t, t+\delta t+1)$, the above procedure is repeated by predicting the locations of nodes starting from time instant $t+\delta t+1$. The sequence of paths obtained under this approach will be denoted as Stable-Mobile-Path$^{\text{Uncertain-Pred}}$ in order to distinguish from the Stable Mobile Path generated when future topology changes are completely known.

3. Simulation study of algorithm *OptPathTrans*

3.1 Simulation conditions

We ran our simulations with a square topology of dimensions 1000m x 1000m. The wireless transmission range of a node is 250m. The node density is varied by performing the simulations in this network with 50 (10 neighbors per node) and 150 nodes (30 neighbors per node). Note that, two nodes *a* and *b* are assumed to have a bidirectional link at time *t* if the Euclidean distance between them at time *t* (derived using the locations of the nodes from the mobility trace file) is less than or equal to the wireless transmission range of the nodes. We obtain a centralized view of the network topology by generating mobility trace files for 1000 seconds in the *ns*-2 network simulator (Bresalu et. al., 2000; Fall & Varadhan, 2001). Each data point in Fig. 6, 7, 8 and 9 is an average computed over 10 mobility trace files and 15 randomly selected *s-d* pairs from each of the mobility trace files. The starting time of each *s-d* session is uniformly randomly distributed between 1 to 20 seconds. The topology sampling interval to generate the mobile graph is 1 second.

3.2 Mobility model

We use the Random Waypoint mobility model (Betstetter et. al., 2004), one of the most widely used mobility simulating models for MANETs. According to this model, each node starts moving from an arbitrary location to a randomly selected destination with a randomly chosen speed in the range [v_{min} ... v_{max}]. Once the destination is reached, the node stays there for a pause time and then continues to move to another randomly selected destination with a different speed. We use v_{min} = 0 and pause time of a node is 0. The values of v_{max} used are 10 and 15 m/s (representing low mobility scenarios), 20 and 30 m/s (representing moderate mobility scenarios), 40 and 50 m/s (representing high mobility scenarios).

3.3 Performance metrics

The performance metrics evaluated are the number of route transitions and the time averaged hop count of the mobile path under the conditions described above. The time averaged hop count of a mobile path is the sum of the products of the number of hops per static path and the number of seconds each static path exists divided by the number of static graphs in the mobile graph. For example, if a mobile path spanning over 10 static graphs comprises of a 2-hop static path p_1, a 3-hop static path p_2, and a 2-hop static path p_3, with each existing for 2, 3 and 5 seconds respectively, then the time-averaged hop count of the mobile path would be (2*2 + 3*3 + 2*5) / 10 = 2.3.

3.4 Obtaining minimum hop mobile path

To obtain the Minimum Hop Mobile Path for a given simulation condition, we adopt the following procedure: When a minimum-hop path is required at time instant *t* and stability is not to be considered, the minimum-hop path Dijkstra algorithm is run on static graph at time instant *t*, and the minimum-hop path obtained is used as long as it exists. We repeat the above procedure until the end of the simulation time.

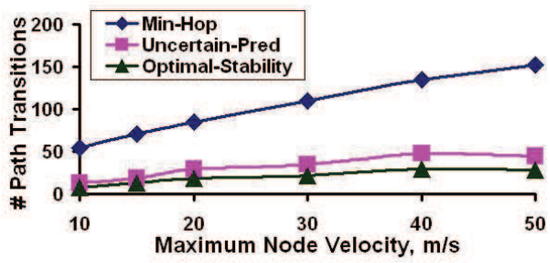


Fig. 6. Stability of Routes (50 Nodes)

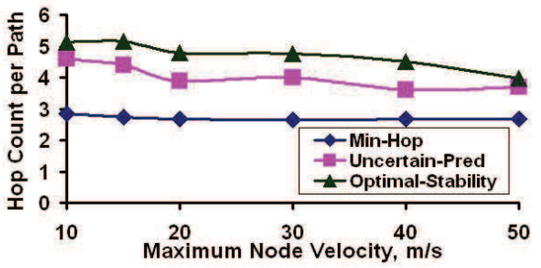


Fig. 7. Hop Count of Routes (50 Nodes)

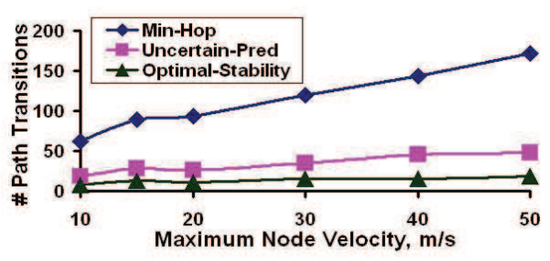


Fig. 8. Stability of Routes (150 Nodes)

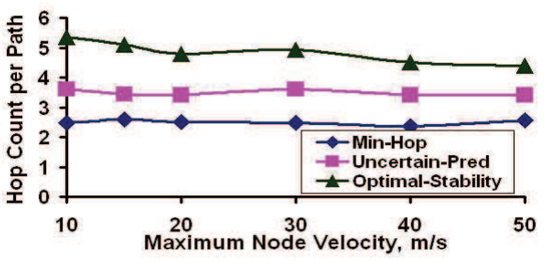


Fig. 9. Hop Count of Routes (150 Nodes)

3.5 Stability-hop count tradeoff

For all simulation conditions, the Minimum Hop Mobile Path incurs the maximum number of route transitions, while the average hop count per Minimum Hop Mobile Path is the least. On the other hand, the Stable Mobile Path incurs the minimum number of route transitions, while the average hop count per Stable Mobile Path is the maximum. The number of route transitions incurred by a Minimum Hop Mobile Path is 5 to 7 times to that of the optimal number of route transitions for a low-density network (refer Fig. 6) and 8 to 10 times to that of the optimal for a high-density network (refer Fig. 8). The average hop count per Stable Mobile Path is 1.5 to 1.8 times to that of the optimal hop count incurred in a low-density network (refer Fig. 7) and is 1.8 to 2.1 times to that of the optimal in a high-density network (refer Fig. 9). Optimality in both these metrics cannot be obtained simultaneously.

3.6 Impact of physical hop distance

The probability of a link (i.e., hop) failure increases with increase in the physical distance between the constituent nodes of the hop. We observed that the average physical distance between the constituent nodes of a hop at the time of a minimum-hop path selection is 70-80% of the transmission range of the nodes, accounting for the minimum number of intermediate nodes to span the distance between the source and destination nodes of the path. On the other hand, the average physical distance between the constituent nodes of a hop at the time of a stable path selection is only 50-55% of the transmission range of the nodes. Because of the reduced physical distance between the constituent nodes of a hop, more intermediate nodes are required to span the distance between the source and destination nodes of a stable path. Hence, the probability of failure of a hop in a stable path is far less compared to that of the probability of a failure of a hop in a minimum hop path. Also, the number of hops does not increase too much so that the probability of a path failure increases with the number of hops. Note that when we have a tie among two or more static paths that have the longest lifetime in a mobile sub graph, we choose the static path that has the minimum hop count to be part of the Stable Mobile Path.

3.7 Impact of node density

As we increase the node density, there are more neighbors per node, which increases the probability of finding a neighbor that is farther away. This helps to reduce the number of hops per path, but the probability of failure of the hop (due to the constituent nodes of the hop moving away) is also high. Thus, for a given value of v_{max}, minimum hop paths are more stable in low-density networks compared to high-density networks (compare Fig. 6 and Fig. 8). The average hop count of a Minimum Hop Mobile Path is more in a low-density network compared to that incurred in a high-density network (compare Fig. 7 and Fig. 9).

When we aim for stable *s-d* paths, we target paths that have low probability of failure due to the constituent nodes of a hop in the path moving away. With increase in node density, algorithm *OptPathTrans* gets more options in selecting the paths that can keep the source and destination connected for a longer time. In high density networks, we have a high probability of finding links whose physical distance is far less than the transmission range of the nodes. This is explored to the maximum by algorithm *OptPathTrans* and hence we observe a reduction in the number route transitions accompanied by an increase in the hop count in high-density networks compared to low-density networks.

3.8 Performance under the prediction with uncertainty model

The number of route transitions incurred by Stable-Mobile-Path$^{\text{Uncertain-pred}}$ is only at most 1.6 to 1.8 times that of the optimal for low-density networks (refer Fig. 6) and 2 to 3 times that of optimal for high-density networks (refer Fig. 8). Nevertheless, the average hop count incurred by Stable-Mobile-Path$^{\text{Uncertain-pred}}$ is 1.3–1.6 times to that incurred by Minimum-Hop-Mobile-Path (refer Fig. 7 and Fig. 9).

The mobility prediction model is practically feasible because the current location of each node, its direction and velocity can be recorded in the Route Request (RREQ) packets that get propagated from the source to destination during an on-demand route discovery. Rather than just arbitrarily choosing a minimum hop path traversed by the RREQ packet and sending a Route Reply (RREP) packet along that path, the destination node can construct a mobile sub graph by incorporating the locations of nodes in the near future, apply algorithm *OptPathTrans*, obtain the Stable-Mobile-Path$^{\text{Uncertain-Pred}}$ and send the RREP along that path.

4. Algorithm for the optimal number of multicast tree transitions

MANETs are deployed in applications such as disaster recovery, rescue missions, military operations in a battlefield, conferences, crowd control, outdoor entertainment activities, etc. One common feature among all these applications is one-to-many multicast communications among the participants. Multicasting is more advantageous than multiple unicast transmissions of the same data independently to each and every receiver, which also leads to network clogging. Hence, to support these applications in dynamic environments like MANETs, ad hoc multicast routing protocols that find a sequence of stable multicast trees are required.

4.1 Multicast steiner tree

Given a weighted 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) $S \subseteq V$, the Steiner tree is the minimum-weight tree of G connecting all the vertices of S. In this chapter, we assume unit weight edges and that all the edges of the Steiner tree are contained in the edge set of the graph. Accordingly, we define the minimum Steiner tree as the tree with the least number of edges required to connect all the vertices in the multicast group (the set of Steiner points). Unfortunately, the problem of determining a minimum Steiner tree in an undirected graph like that of the unit disk graph is NP-complete. Efficient heuristics (e.g., Kou et. al., 1981) have been proposed in the literature to approximate a minimum Steiner tree.

4.2 Stable mobile multicast steiner tree vs minimum mobile multicast steiner tree

Aiming for the minimum Steiner tree in MANETs, results in multicast trees that are highly unstable. The multicast tree has to be frequently rediscovered, and this adds considerable overhead to the resource-constrained network. By adding a few more links and nodes to the tree, it is possible to increase its stability. We define stability of a multicast Steiner tree in terms of the number of times the tree has to change for the duration of a multicast session. Extending the greedy approach of *OptPathTrans* to multicasting, we propose an algorithm called *OptTreeTrans* to determine the minimum number of tree transitions incurred during the period of a multicast session for a multicast group comprising of a source node and a set of receiver nodes. Given the complete knowledge of future topology changes, the algorithm

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 Steiner trees obtained by running this algorithm is called the Stable Mobile Multicast Steiner Tree. We use the Kou. et. al's (Kou et. al., 1981) well-known $O(|V||S|^2)$ heuristic, as the underlying heuristic to determine the longest existing multicast Steiner tree. A Minimum Mobile Multicast Steiner Tree is the sequence of approximations to the minimum Steiner tree obtained by directly using Kou's heuristic whenever required.

4.3 Heuristic to approximate minimum steiner tree

We use the Kou et. al's (Kou et. al., 1981) well-known $O(|V||S|^2)$ heuristic ($|V|$ is the number of nodes in the network graph and $|S|$ is the size of the multicast group) to approximate the minimum Steiner tree in graphs representing snapshots of the network topology. We give a brief outline of the heuristic in Fig. 10. An (s-S)-tree is defined as the multicast Steiner tree connecting a source node s to all the members of the multicast group S, which is also the set of Steiner points. Note that $s \in S$.

Input: An undirected graph $G = (V, E)$
Multicast group $S \subseteq V$
Output: A tree T_H for the set S in G

Step 1: Construct a complete undirected weighted graph $G_C = (S, E_C)$ from G and S where $\forall (v_i, v_j) \in E_C$, v_i and v_j are in S, 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_S 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_S in G_S (If more than one minimal spanning tree exists, pick an arbitrary one). Note that each edge in G_S has weight 1.
Step 5: Construct the minimum Steiner tree T_H, from T_S by deleting edges in T_S, if necessary, such that all the leaves in T_H are members of S.

Fig. 10. Kou et. al's Heuristic (Kou et. al., 1981) to find an Approximate Minimum Steiner Tree

4.4 Algorithm *OptTreeTrans*

Let $G_M = G_1G_2 \ldots G_T$ 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-S)-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-S)-tree in $G(i, j)$ and none exists in $G(i, j+1)$. A multicast (s-S)-tree in $G(i, j)$ is selected using Kou's heuristic. Such a tree exists in each of the static graphs $G_i, G_{i+1}, \ldots, G_j$. If there is a tie, the (s-S)-tree with the smallest number of constituent links is chosen. If sampling instant $t_{j+1} \leq t_T$, the above procedure is repeated by

finding the (*s*-*S*)-tree that can survive for the maximum amount of time since t_{j+1}. A sequence of such maximum lifetime multicast Steiner (*s*-*S*) trees over the timescale of G_M forms the Stable Mobile Multicast Steiner Tree in G_M. The pseudo code is given in Fig. 11.

Input: $G_M = G_1G_2 \ldots G_T$, source *s*, multicast group *S*
Output: $(s\text{-}S)_{MobileStabletree}$ // Stable-Mobile-Multicast-Steiner-Tree
Auxiliary Variables: *i*, *j*
Initialization: $i=1$; $j=1$; $(s\text{-}S)_{MobileStabletree} = \Phi$

Begin *OptTreeTrans*

1 **while** (i ≤ 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*-*S*)-tree in $G(i, j)$ and { no (*s*-*S*)-tree exists in $G(i, j+1)$ or $j = T$ }

3 $(s\text{-}S)_{MobileStabletree} = (s\text{-}S)_{MobileStabletree}$ U {Minimum Steiner (*s*-*S*)-tree in $G(i, j)$ }

4 $i = j + 1$

5 end while

6 **return** $(s\text{-}S)_{MobileStabletree}$

End *OptTreeTrans*

Fig. 11. Pseudo Code for Algorithm *OptTreeTrans*

4.5 Algorithm complexity and proof of correctness

In a mobile graph $G_M = G_1G_2 \ldots G_T$, the number of tree transitions can be at most *T*. The minimum Steiner tree Kou's heuristic will have to be run at most *T* times, each time on a graph of $|V|$ nodes. As Kou's heuristic is of $O(|V||S|^2)$ worst-case run-time complexity where $|S|$ is the size of the multicast group, the worst-case run-time complexity of *OptTreeTrans* is $O(|V||S|^2T)$.

Given a mobile graph $G_M = G_1G_2 \ldots G_T$, source node *s* and multicast group *S*, let the number of tree transitions in the Mobile Multicast Steiner Tree, $(s\text{-}S)_{MobileStabletree}$, generated by *OptTreeTrans* be *m*. To prove *m* is optimal, assume the contrary: there exists another Mobile Multicast Steiner Tree $(s\text{-}S)'_{MobileStabletree}$ in G_M and the number of tree transitions in $(s\text{-}S)'_{MobileStabletree}$ is $m' < m$.

Let $epoch_S^1, epoch_S^2, \ldots, epoch_S^m$ be the set of sampling time instants in each of which the Mobile Multicast Steiner Tree $(s\text{-}S)_{MobileStabletree}$ suffers no tree transitions. Let $epoch_{S'}^1$, $epoch_{S'}^2$, …, $epoch_{S'}^{m'}$ be the set of sampling time instants in each of which the Mobile Multicast Steiner Tree $(s\text{-}S)'_{MobileStabletree}$ suffers no tree transitions. Let $t_{(s-S)}^{init,j}$ and $t_{(s-S)}^{end,j}$ be the

initial and final sampling time instants of $epoch_S^j$ where $1 \le j \le m$. Let $t_{(s-S)'}^{init,k}$ and $t_{(s-S)'}^{end,k}$ be the initial and final sampling time instants of $epoch_{S'}^k$ where $1 \le k \le m'$. Note that $t_S^{init,1} = t_{S'}^{init,1}$ and $t_S^{end,m} = t_{S'}^{end,m'}$ to indicate $(s\text{-}S)_{MobileStabletree}$ and $(s\text{-}S)'_{MobileStabletree}$ span over the same time period, T, of the network session.

Now, since we claim that $m' < m$, there should exist j, k where $1 \le j \le m$ and $1 \le k \le m'$ such that $epoch_S^j \subset epoch_{S'}^k$, i.e., $t_{S'}^{init,k} < t_S^{init,j} < t_S^{end,j} < t_{S'}^{end,k}$ and at least one $(s\text{-}S)'$-tree existed in $[t_{S'}^{init,k},\ldots,t_{S'}^{end,k}]$. In other words, there should be at least one $(s\text{-}S)'$-tree in $(s\text{-}S)'_{MobileStabletree}$ that has a lifetime larger than that of the lifetime of the $(s\text{-}S)$-trees in $(s\text{-}S)_{MobileStabletree}$. But, algorithm *OptTreeTrans* made a tree transition at $t_S^{end,j}$ since there was no $(s\text{-}S)$-tree from $t_S^{init,j}$ beyond $t_S^{end,j}$. Thus, there is no $(s\text{-}S)$-tree in the range $[t_S^{init,j}, \ldots, t_{S'}^{end,k}]$ and hence there is no $(s\text{-}S)$-tree in the range $[t_{S'}^{init,k},\ldots,t_{S'}^{end,k}]$. This shows that the lifetime of each of the $(s\text{-}S)'$-trees in $(s\text{-}S)'_{MobileStabletree}$ has to be smaller or equal to the lifetime of the $(s\text{-}S)$-trees in $(s\text{-}S)_{MobileStabletree}$, implying $m' \ge m$. This is a contradiction and proves that our hypothesis $m' < m$ is not correct. Hence, the number of tree transitions in $(s\text{-}S)_{MobileStabletree}$ is optimal and $(s\text{-}S)_{MobileStabletree}$ is the Stable Mobile Multicast Steiner Tree.

4.6 Example run of algorithm *OptTreeTrans*

Consider the mobile graph $G_M = G_1G_2G_3G_4G_5$ sampled every second (Fig. 12). Let node 1 be the source node and nodes 5 and 6 be the receivers of the multicast group. The Minimum Mobile Steiner Tree in G_M is {{1-3, 3-6, 5-6}G_1, {1-4, 4-6, 4-5}G_2, {1-2, 2-6, 5-6}G_3, {1-3, 3-6, 5-6}G_4, {1-2, 2-6, 2-5}G_5}. The edges of the constituent minimum Steiner trees in each of the static graphs are shown in dark lines. The number of tree transitions is 5 and the time averaged number of edges per Minimum Mobile Steiner Tree is 3 as there are three edges in each constituent minimum Steiner tree. The execution of algorithm *OptTreeTrans* on the mobile graph G_M is shown in Fig. 13. The Stable Mobile Steiner Tree formed is {{1-4, 4-3, 3-6, 4-5}G_{12}, {1-2, 2-3, 3-6, 2-4, 4-5}G_{345}}. The number of tree transitions is 2 and the time-averaged number of edges in the Stable Mobile Steiner Tree is (4*2 + 5*3)/5 = 4.6 as there are 4 edges in the stable Steiner tree common to graphs G_1 and G_2 and 5 edges in the stable Steiner tree common to G_3, G_4 and G_5. The simulation results also vindicate such tradeoff between the number of Steiner tree transitions and number of edges in the mobile Steiner tree.

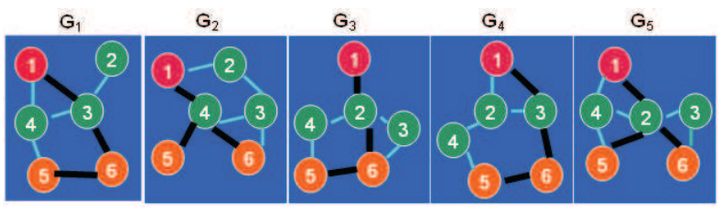


Fig. 12. Mobile Graph and Minimum-Mobile-Steiner-Tree

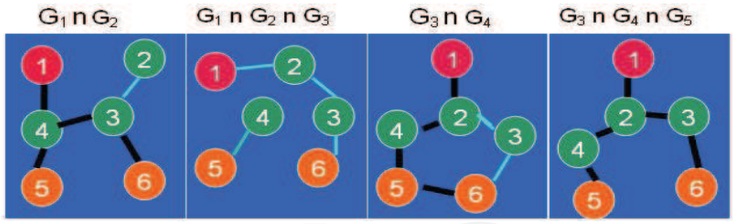


Fig. 13. Execution of Algorithm *OptTreeTrans* on Fig. 12.

5. Simulation study of algorithm *OptTreeTrans*

5.1 Simulation conditions

We ran our simulations with a square topology of dimensions 1000m x 1000m. The wireless transmission range of a node is 250m. The node density is varied by performing the simulations in this network with 50 (10 neighbors per node) and 150 nodes (30 neighbors per node). We obtain a centralized view of the network topology by generating mobility trace files for 1000 seconds in *ns*-2 (Bresalu et. al., 2000; Fall & Varadhan, 2001). Random waypoint mobility model is the mobility model used in the simulations. The range of node velocity values used are [0...10 m/s] and [0...50 m/s]. Each multicast group includes a source node and a set of receiver nodes. The multicast group size values are 2, 4, 8, 12, 18 and 24. Each data point in Fig. 14 through 21 is an average computed over 10 mobility trace files and 5 randomly selected groups for each size. The starting time of each multicast session is uniformly randomly distributed between 1 to 50 seconds and the simulation time is 1000 seconds. The topology sampling interval to generate the mobile graph is 1 second.

5.2 Minimum mobile multicast steiner tree and performance metrics

When an (*s*-*S*) Steiner tree is required at sampling time instant t_i and stability is not to be considered, then Kou's heuristic is run on static graph G_i and the (*s*-*S*) tree obtained is used as long as it exists. The procedure is repeated till the last sampling time instant t_T is reached. We refer to the sequence of multicast Steiner trees generated by the above strategy as Minimum Mobile Multicast Steiner Tree. The performance metrics evaluated are the number of tree transitions and the average number of edges in the mobile Steiner trees, which is the number of links in the constituent (*s*-*S*) Steiner trees, averaged over time.

5.3 Minimum mobile multicast steiner tree vs stable mobile multicast steiner tree

The number of multicast tree transitions increases rapidly with increase in multicast group size (refer Fig. 14, 16, 18 and 20). On the other hand, by accommodating 10-40% more edges (refer Fig. 15, 17, 19 and 21), stability of the Stable Mobile Multicast Steiner Tree is almost insensitive to multicast group size. For given value of v_{max}, the number of tree transitions incurred by the Minimum Mobile Multicast Steiner Tree in a low-density network (refer Fig. 14 and 18) is 5 (with group size of 4) to 10 (with group size of 24) times to that of the optimal. In high-density networks (refer Fig. 16 and 20), the number of tree transitions incurred by the Minimum Mobile Multicast Steiner Tree is 8 (with group size of 4) to 25 (with group size of 24) times to that of the optimal.

For a given node mobility and multicast group size, as the network density increases, algorithm *OptTreeTrans* makes use of the available nodes and links as much as possible in order to maximize the stability of the trees. For a Minimum Mobile Steiner Tree, the average

number of links in the constituent (*s*-*S*) trees is the same with increase in node density; the stability of the Minimum Mobile Steiner Trees decreases with increase in node density.

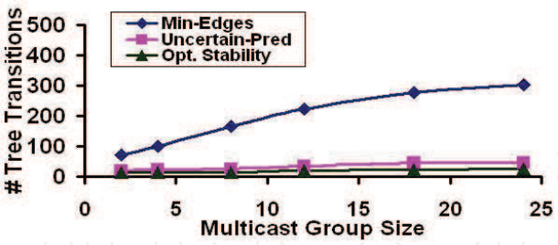


Fig. 14. Stability of Trees
(50 Nodes, v_{max} = 10 m/s)

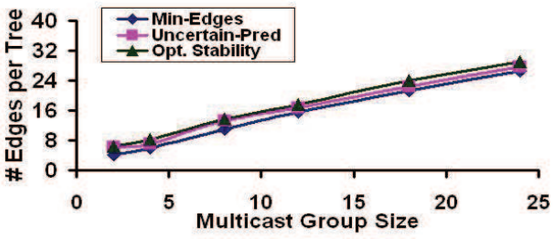


Fig. 15. Edges per Tree
(50 Nodes, v_{max} = 10 m/s)

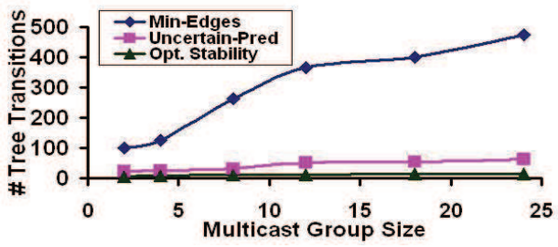


Fig. 16. Stability of Trees
(150 Nodes, v_{max} = 10 m/s)

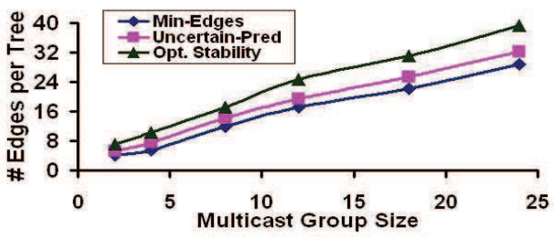


Fig. 17. Edges per Tree
(150 Nodes, v_{max} = 10 m/s)

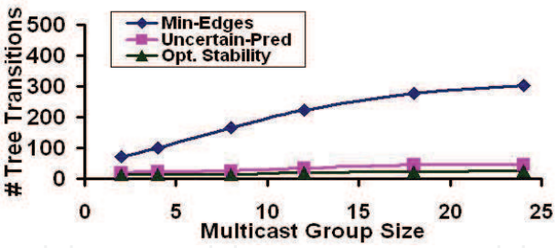


Fig. 18. Stability of Trees
(50 Nodes, v_{max} = 50 m/s)

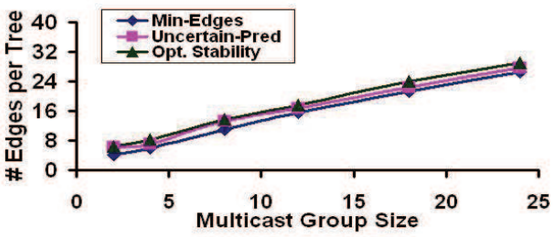


Fig. 19. Edges per Tree
(50 Nodes, v_{max} = 50 m/s)

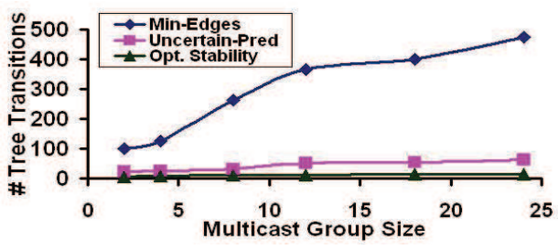


Fig. 20. Stability of Trees
(150 Nodes, v_{max} = 50 m/s)

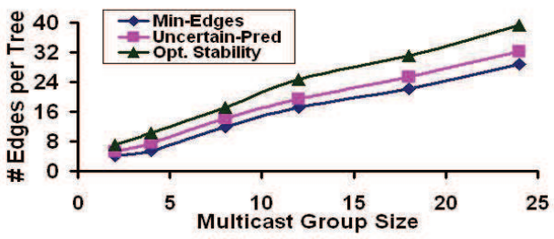


Fig. 21. Edges per Tree
(150 Nodes, v_{max} = 50 m/s)

For a given value of v_{max}, the number of tree transitions incurred by the Stable-Mobile-Multicast-Steiner-Tree$^{Uncertain\text{-}pred}$ in a low-density network (refer Fig. 14 and 18) is 1.6 (with group size of 2) to 2 (with group size of 24) times to that of the optimal, while in a high-density network (refer Fig. 16 and 20), the number of tree transitions is 3 (with group size of 2) to 4 (with group size of 24) times to that of the optimal. Thus, the stability of the constituent static trees in Stable-Mobile-Multicast-Steiner-Tree$^{Uncertain\text{-}pred}$ is not much affected by multicast group size and the number of edges (refer Fig. 15, 17, 19 and 21) is at most 40% more than that found in a Minimum Mobile Multicast Steiner Tree.

6. Impact of the stability-hop count tradeoff on network resources and routing protocol performance

We now discuss the impact of the stability-hop count tradeoff on network resources like the available node energy and the performance metrics like end-to-end delay per data packet.

6.1 Energy consumption

With increase in network density, a Stable Mobile Path incurs a reduced number of route transitions at the cost of an increased hop count; while a Minimum Hop Mobile Path incurs a reduced hop count per path at the cost of an increase in the number of route transitions. In (Meghanathan & Farago, 2006a), we analyzed the impact of these contradicting route selection policies on the overall energy consumption of a source-destination (*s*-*d*) session when using a Stable Mobile Path vis-à-vis a Minimum Hop Mobile Path for on-demand routing in MANETs. Some of the significant observations include:

- As we reduce the energy consumed per hop for data packet transfer (i.e., as we adopt reduced overhearing or no overhearing models), a Stable Mobile Path can bring significant energy savings than that obtained using a Minimum Hop Mobile Path.
- When data packets are sent continuously but at a reduced rate, we should use stable paths. If minimum hop paths are used, we may end up discovering a route to send every data packet, nullifying the energy savings obtained from reduced hop count.
- At high data packet rates (i.e., high data traffic), even a slight increase in the hop count can result in high energy consumption, especially in the presence of complete overhearing (also called as promiscuous listening). At high data traffic, energy spent in route discovery is overshadowed by the energy spent in data packet transfer.
- Route discovery is very expensive with respect to energy consumption in networks of high density compared to networks of low density. To minimize the overall energy consumption at moderate data traffic, we should use minimum-hop based routing at low network densities and stability-based routing at high network densities.

6.2 End-to-end delay per data packet

In (Meghanathan, 2008), we studied the performance of stable path routing protocols like ABR, FORP and RABR in *ns*-2 and measured the route stability and the end-to-end delay per data packet for *s*-*d* sessions running each of these three protocols. We observed a stability-hop count tradeoff within the class of stability-based routing protocols and the three protocols are ranked in the following increasing order of hop count: ABR, RABR and FORP; while in terms of the increasing order of the number of route transitions per *s*-*d* session, the ranking is: FORP, RABR and ABR. At low and moderate mobility conditions

(v_{max} <= 30 m/s), ABR routes incurred the lowest delay per packet compared to that of FORP. This could be attributed to the higher route relaying load on the nodes. Especially at high data traffic load, FORP routes incur significant delays due to MAC layer contention and queuing before transmission. RABR achieves a right balance between the route relaying load per node and the route discovery latency. RABR routes incur an end-to-end delay per packet that is close to that of ABR at low and moderate velocities and at the same time achieve stability close to that of the FORP routes. At high velocity, the buffering delay due to the route acquisition latency plays a significant role in increasing the delay of ABR routes and to a certain extent the RABR routes. Thus, at high node mobility conditions, all the three protocols incur end-to-end delay per packet that is close enough to each other.

7. Conclusions and future work

In this chapter, we described algorithms *OptPathTrans* and *OptTreeTrans* to determine respectively the sequence of stable paths (Stable Mobile Path) and multicast trees (Stable Mobile Multicast Steiner Tree) over the duration of a MANET session. Performance study of the two algorithms, when the complete knowledge of future topology changes is available at the time of path/tree selection, illustrates a distinct tradeoff between path hop count and the number of path transitions, and the number of edges in the multicast Steiner tree and the number of multicast Steiner tree transitions. It is highly impossible to simultaneously achieve optimality in the above mentioned contrasting performance metrics for paths and trees. The sequence of stable paths and trees generated by the two algorithms under the "Prediction with Uncertainty" model are highly stable compared to their minimum mobile versions. Also, the hop count, the number of edges and the number of nodes in the stable paths and trees is not as high as that observed in the stable mobile paths and trees obtained when the algorithms are run with complete knowledge of the future topology changes.

Note that the Dijkstra algorithm and the Kou et. al heuristic are merely used as a tool to find the appropriate stable communication structures. The optimal number of route and tree reconstructions does not depend on these underlying algorithms as we try to find the longest living route and tree in the mobile sub graph spanning a sequence of static graphs. But, the run-time complexity of the two algorithms depends on the underlying algorithm used to determine the Stable Mobile Path and the Stable Mobile Multicast Steiner Tree.

Future work is on the following: (i) To develop distributed versions of *OptPathTrans* and *OptTreeTrans* by extending these algorithms respectively as unicast and multicast routing protocols, (ii) To study the performance of algorithms *OptPathTrans* and *OptTreeTrans* under other MANET mobility models like Random Walk, Random Direction and Gauss-Morkov models (Camp et. al., 2002) and (iii) To develop various location-update and mobility prediction mechanisms to gather and/or distribute knowledge of future topology changes.

8. References

Agarwal, S.; Ahuja, A.; Singh, J. P. & Shorey, R. (2000). Route-Life Time Assessment Based Routing Protocol for Mobile Ad hoc Networks, *Proceedings of the IEEE International Conference on Communications*, pp. 1697-1701, ISBN: 0780362837, June 2000, New Orleans, LA, USA.

Bettstetter, C.; Hartenstein, H. & Perez-Costa, X. (2004). Stochastic Properties of the Random Way Point Mobility Model. *Wireless Networks*, Vol. 10, No. 5, (September 2004), pp. 555-567, ISSN: 10220038.

Breslau, L.; Estrin, D.; Fall, K.; Floyd, S.; Heidemann, J.; Helmy, A.; Huang, P.; McCanne, S.; Varadhan, K.; Xu, Y.; Yu, H. (2000). Advances in Network Simulation. *IEEE Computer*, Vol. 33, No. 5 (May 2000), pp. 59-67, ISSN: 00189162.

Camp, T.; Boleng, J. & Davies, V. (2002). A Survey of Mobility Models for Ad Hoc Network Research, *Wireless Communication and Mobile Computing*, Vol. 2, No. 5, (September 2002), pp. 483-502, ISSN: 15308669.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L. & Stein, C. (2001). *Introduction to Algorithms*, 2nd Edition, MIT Press, ISBN: 0262032937.

Corson, M. S. & Ephremides, A. (1995). A Distributed Routing Algorithm for Mobile Wireless Networks. *Wireless Networks*, Vol. 1, No. 1, (February 1995), pp. 61-81, ISSN: 10220038.

Fall, K. & Varadhan, K. (2001). ns notes and documentation. The VINT Project at LBL, Xerox PARC, UCB, and USC/ISI, http://www.isi.edu/nsnam/ns, August 2001.

Farago, A. & Syrotiuk, V. R. (2003). MERIT: A Scalable Approach for Protocol Assessment. *Mobile Networks and Applications*, Vol. 8, No. 5, (October 2003), pp. 567 – 577, ISSN: 1383-469X.

Johnson, D. B.; Maltz, D. A. & Broch, J. (2001). DSR: The Dynamic Source Routing Protocol for Multi-hop Wireless Ad hoc Networks, In: Ad hoc Networking, Charles E. Perkins, (Ed.), 139 – 172, Addison Wesley, ISBN: 0201309769.

Kou, L.; Markowsky, G. & Berman, L. (1981). A Fast Algorithm for Steiner Trees. *Acta Informatica*, Vol. 15, No. 2, (June 1981), pp. 141-145, ISSN: 0001-5903.

Meghanathan, N. & Farago, A. (2004). Survey and Taxonomy of 55 Unicast Routing Protocols for Mobile Ad hoc Networks, Technical Report UTD-CSC-40-04, The University of Texas at Dallas, Richardson, TX, November 2004.

Meghanathan, N. & Farago, A. (2005). An Efficient Algorithm for the Optimal Number of Route Transitions in Mobile Ad hoc Networks. *Proceedings of the 1st IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, Vol. 3, pp. 41-48, ISBN: 0780391810, August 2005, Montreal, Canada.

Meghanathan, N. & Farago, A. (2006a). Comparison of Routing Strategies for Minimizing Energy Consumption in Mobile Ad Hoc Networks. *Proceedings of the 4th Asian International Mobile Computing Conference*, pp. 3-11, ISBN: 0070608342, January 2006, Kolkatta, India.

Meghanathan, N. (2006b). An Algorithm to Determine the Sequence of Stable Connected Dominating Sets in Mobile Ad Hoc Networks, *Proceedings of 2nd Advanced International Conference on Telecommunications*, ISBN: 0769525229, February 2006, Guadeloupe, French Caribbean.

Meghanathan, N. (2006c). Determining a Sequence of Stable Multicast Steiner Trees in Mobile Ad hoc Networks. *Proccedings of the 44th ACM Southeast Conference*, pp. 102-106, ISBN: 1595933158, March 2006, Melbourne, FL, USA.

Meghanathan, N. (2006d). A Simulation Study on the Stability-Oriented Routing Protocols for Mobile Ad hoc Networks. *Proceedings of the IFIP International Conference on*

Wireless and Optical Communication Networks, ISBN: 1424403405, April 2006, Bangalore, India.

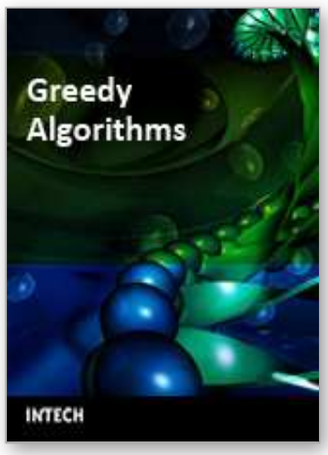
Meghanathan, N. (2007). Path Stability based Ranking of Mobile Ad hoc Network Routing Protocols. *ISAST Transactions Journal on Communications and Networking*, Vol. 1, No. 1, (August 2007), pp. 66-73, ISSN: 17970989.

Meghanathan, N. (2008). Exploring the Stability-Energy Consumption-Delay-Network Lifetime Tradeoff of Mobile Ad hoc Network Routing Protocols. Journal of Networks, Vol. 3, No. 2, (February 2008), pp. 17-28, ISSN: 17962056.

Perkins, C. E. & Royer, E. M. (1999). Ad hoc On-demand Distance Vector Routing, *Proceedings of the Second Annual IEEE International Workshop on Mobile Computing Systems and Applications*, pp. 90–100, ISBN: 0769500250, February 1999, New Orleans, LA, USA.

Su, W.; Lee, S.-J. & Gerla, M. (2001). Mobility Prediction and Routing in Ad hoc Wireless Networks. *International Journal of Network Management*, Vol. 11, No. 1, (Jan-Feb. 2001), pp. 3-30, ISSN: 10991190.

Toh, C.-K. (1997). Associativity-Based Routing for Ad hoc Mobile Networks. *IEEE Personal Communications*, Vol. 4, No. 2, (March 1997), pp. 103 – 109, ISSN: 10709916.



Greedy Algorithms

Edited by Witold Bednorz

ISBN 978-953-7619-27-5
Hard cover, 586 pages
Publisher InTech
Published online 01, November, 2008
Published in print edition November, 2008

Each chapter comprises a separate study on some optimization problem giving both an introductory look into the theory the problem comes from and some new developments invented by author(s). Usually some elementary knowledge is assumed, yet all the required facts are quoted mostly in examples, remarks or theorems.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Natarajan Meghanathan (2008). Greedy Algorithms to Determine Stable Paths and Trees in Mobile Ad hoc Networks, Greedy Algorithms, Witold Bednorz (Ed.), ISBN: 978-953-7619-27-5, InTech, Available from: http://www.intechopen.com/books/greedy_algorithms/greedy_algorithms_to_determine_stable_paths_and_trees_in_mobile_ad_hoc_networks

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen