On the stability of paths, Steiner trees and connected dominating sets in mobile ad hoc networks

Natarajan Meghanathan a,*, Andras Farago b

a Department of Computer Science, Jackson State University, 1400 J.R. Lynch Street, Jackson, MS 39217, USA
b Department of Computer Science, University of Texas at Dallas, USA

Received 27 August 2005; received in revised form 27 May 2007; accepted 26 June 2007
Available online 1 July 2007

Abstract

We propose algorithms that use the complete knowledge of future topology changes to set up benchmarks for the minimum number of times a communication structure (like paths, trees, connected dominating sets, etc.) should change in the presence of a dynamically changing topology. We first present an efficient algorithm called OptPathTrans that operates on a simple greedy principle: whenever a new source–destination (s–d) path is required at time instant t, choose the longest-living s–d path from time t. The above strategy when repeated over the duration of the s–d session yields a sequence of long-lived stable paths such that number of path transitions is the global minimum. We then propose algorithms to determine the sequence of stable Steiner trees and the sequence of stable connected dominating sets to illustrate that the principle behind OptPathTrans is very general and can be used to find the stable sequence of any communication structure as long as there is a heuristic or algorithm to determine that particular communication structure in a given network graph. We study the performance of the three algorithms in the presence of complete knowledge of future topology changes as well as using models that predict the future locations of nodes. Performance results indicate that the stability of the communication structures could be considerably improved by making use of the knowledge about locations of nodes in the near future.

© 2007 Elsevier B.V. All rights reserved.

Keywords: Mobile ad hoc networks; Stability; Optimal route transitions; Simulation; Steiner trees; Connected dominating sets

1. Introduction

Stability is an important design criterion to be considered while developing multi-hop communication protocols (e.g. routing) for resource-constrained environments like mobile ad hoc networks (MANETs). MANETs are easily prone to congestion due to the low to moderate capacity of the wireless links. Also, mobile nodes used in energy-constrained environments like sensor networks and embedded networks cannot afford to lose their battery power quickly. Protocols that often change their communication structures (e.g., paths, trees, connected dominating set, etc.) to maintain optimality in their design metric (e.g., hop count, delay) incur a huge network overhead. For example, the commonly used route discovery approach of
flooding the route request and/or reply packets can easily lead to congestion and also consume battery power. Frequent route changes can also result in out-of-order 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 a quality of service (QoS) point of view.

A majority of the MANET routing protocols (e.g., DSR [11], AODV [19]) were proposed to optimize one or more performance metric in a greedy fashion without looking at the future [14,16]. To maintain optimality in their design metrics, routing protocols change their paths frequently and incur a huge network overhead. Very few routing protocols that target stability of routes have been proposed in literature: associativity-based routing (ABR) [21] selects paths based on the degree of association stability, which basically quantifies how long two nodes have stayed together as neighbors. Flow-oriented routing protocol (FORP) [20] 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. Route-lifetime assessment based routing (RABR) protocol [1] uses the average change in the received signal strength to predict the time when the received signal strength would fall below a critical threshold. These stable path MANET routing protocols are distributed and on-demand in nature and thus are not guaranteed to determine the most stable routes.

We could not find any work that deals with determining the sequence of stable routes (and hence the minimum number of route transitions) for a source–destination (s–d) communication session. This formed the motivation for our paper. In the first half of the paper, we address the issue of finding a sequence of stable routes such that the number of route transitions is the global minimum. In this context, we quantify path stability in terms of the number of route transitions a routing protocol incurs to continue the data exchange between a particular s–d pair. We present a simple but powerful polynomial-time greedy algorithm OptPathTrans to determine the minimum required number of route transitions. Given the complete knowledge of the future topology changes, the algorithm operates on the simple greedy principle: Whenever an s–d path is required at a time instant t, choose the longest-living s–d path from t. The above strategy is repeated over the duration of the s–d session. The sequence of such longest living stable paths is called the Stable-Mobile-Path. Note that in this paper, we use the terms path and route, edge and link interchangeably.

We use the big-O notation to express the theoretical worst-case run-time complexity of the algorithms discussed in this paper. Given a problem size x, where x is usually the number of items, we say \( f(x) = O(g(x)) \), when there exists positive constants c and k such that \( 0 \leq f(x) \leq cg(x) \), for all \( x \geq k \) [7]. The worst-case run-time complexity of OptPathTrans is \( O(n^2T) \), where \( n \) is the number of nodes in the network, \( T \) is the duration of the s–d session. \( O(n^2) \) is the worst-case run-time complexity of Dijkstra algorithm – the path-finding algorithm we use to determine each constituent path of the Stable-Mobile-Path.

In the second and third parts of the paper, we show that the 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 to determine the sequence of stable multicast Steiner trees for a multicast session and algorithm OptCDSTrans to determine the sequence of stable connected dominating sets (CDS) for a network 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 and \( E \) is the set of edges connecting these nodes, and a subset \( S \subseteq V \) of vertices 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. Given an undirected network graph, a connected dominating set (CDS) is a sub-graph of the graph such that all the nodes in the graph are either in the CDS or attached to a node in the CDS. It is very rare that greedy strategies give an optimal solution. Algorithms OptPathTrans, OptTreeTrans and OptCDSTrans join the league of Dijkstra algorithm, Minimum spanning tree Kruskal and Prim algorithms [7] that have used greedy strategies, but yet give optimal solution.

In practical scenarios, it might not be possible to have the complete knowledge of future topology
changes at the time of path/tree/CDS selection. In
addition to studying the performance of the three
algorithms when the complete knowledge of future
topology changes is available, we use the following
two approaches to predict the future topology
changes and observe the behavior of each of the
three algorithms: (i) Predict with Uncertainty – we
predict 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, and (2) Look-ahead – we predict the
future locations of nodes at different time instants
based on the knowledge of future topology changes
available for a limited time-period called look-ahead
window size. Under both these approaches, we
observe that with just using the knowledge of future
topology changes in the near future, it is possible to
find a sequence of paths/ trees/ CDSs such that the
number of transitions is very minimal.

The rest of the paper is organized as follows: In
Section 2, we propose algorithm OptPathTrans to
determine the Stable-Mobile-Path. Section 3 starts
with a review of Kou et al.’s heuristic to approximate
the minimum Steiner tree and then proposes algo-
rithm OptTreeTrans to determine the sequence of
stable Steiner trees for a multicast session. Section
4 starts with a discussion of the d-MCDS heuristic
to approximate a minimum connected dominating
set (MCDS) and then proposes algorithm OptCDS-
Trans to determine the sequence of stable CDS for
a network session. In each of Sections 2–4, we also
discuss the proof of correctness and the run-time com-
plexity of the algorithm proposed in that section. We
analyze the performance of each of these algorithms
with extensive simulations conducted when the
knowledge of future topology changes is (i) com-
pletely available, (ii) based on the look-ahead
approach and (iii) predicted with uncertainty based
on the current location, direction of movement, plus
the velocity of nodes. Section 5 discusses the impact
of the stability-hop count tradeoff on performance
metrics like energy consumption and end-to-end
delay per packet. Section 6 concludes the paper
and discusses future work.

2. Algorithm for optimal number of path transitions

Though resorting to flooding at high mobility
could be considered a viable alternative [8], the
energy consumption and congestion will prohibi-
tively increase with data load and also the privacy
of the source–destination (s–d) pair will be at stake.
This motivates the need for stable path routing
algorithms and protocols in dynamically changing
scenarios, typical to that of MANETs. We now
present an algorithm to find the sequence of stable
paths and hence the minimum number of path transi-

tions for an s–d session.

2.1. Mobile graph

A mobile graph [10] is defined as the sequence
\( G_M = G_1G_2\cdots G_T \) of static graphs that
represents the network topology changes over some time scale
\( T \). In the simplest case, the mobile graph
\( G_M = G_1G_2\cdots G_T \) can be extended by a new instantaneous
graph \( G_{T+1} \) to a longer sequence \( G_M = G_1G_2\cdots G_TG_{T+1} \), 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 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 [10], defined for a source–destina-
tion (s–d) pair, in a mobile graph \( G_M = G_1G_2\cdots G_T \)
is the sequence of paths \( P_M = P_1P_2\cdots P_T \), 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 \( v_0v_1\cdots v_l \), 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 dura-
tion of a session between \( s \) and \( d \). 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
\( P_i = v_0v_1\cdots v_l \) 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 \( G_M = G_1G_2\cdots G_T \) and s–d
pair, the weight of a mobile path \( P_M = P_1P_2\cdots P_T \) is
\[
 w(P_M) = \sum_{i=1}^{T} w_i(P_i) + \sum_{i=1}^{T-1} C_{\text{trans}}(P_i, P_{i+1}) \tag{2}
\]
where \( C_{\text{trans}}(P_i, P_{i+1}) \) 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 \( w(P_i) \).

### 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 Eq. (2), a Stable-Mobile-Path minimizes only the sum of the transition costs \( \sum_{i=1}^{T-1} C_{\text{trans}}(P_i, P_{i+1}) \) and a Minimum-Hop-Mobile-Path minimizes only the term \( \sum_{i=1}^{T} w(P_i) \), 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 \( w(P_M) = \sum_{i=1}^{T} w(P_i) + \sum_{i=1}^{T-1} C_{\text{trans}}(P_i, P_{i+1}) \) has been proposed in [10].

### 2.4. Algorithm description

Algorithm \( \text{OptPathTrans} \) operates on the following greedy strategy: Whenever a path is required, select a path that will exist for the longest time. Let \( G_M = G_1 G_2 \ldots G_T \) be the mobile graph generated by sampling the network topology at regular instants \( t_1, t_2, \ldots, t_T \) of 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 \( t_{i+1} \leq t_T \), then the above procedure is repeated by finding the \( s-d \) path that can survive for the maximum amount of time since \( t_{i+1} \). A sequence of such maximum lifetime static \( s-d \) paths over the timescale of a mobile graph \( G_M \) forms the stable mobile \( s-d \) path in \( G_M \).

The pseudo code of the algorithm is given in Fig. 1.

### 2.5. Algorithm complexity and proof of correctness

In a mobile graph \( G_M = G_1 G_2 \ldots G_T \), 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 \( \text{O}(n^2) \), where \( n \) is the number of nodes in the network, the worst-case run-time complexity of \( \text{OptPathTrans} \) is \( \text{O}(n^2 T) \).

We use the proof by contradiction technique to prove the correctness of algorithm \( \text{OptPathTrans} \). Let \( P_S \) (with \( m \) route transitions) be the mobile path generated by algorithm \( \text{OptPathTrans} \). To prove \( m \) is optimal, we assume the contrary that there exists

---

**Input:** \( G_M = G_i G_2 \ldots G_T \), source \( s \), destination \( d \)  
**Output:** \( P_S \)  
**Auxiliary Variables:** \( i, j \)  
**Initialization:** \( i=1; j=1; P_S=\emptyset \)

**Begin** \( \text{OptPathTrans} \)

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. \( P_S = P_S \cup \{ \text{minimum hop s-d path in } G(i, j) \} \)

4. \( i = j + 1 \)

5. end while

6. return \( P_S \)

**End** \( \text{OptPathTrans} \)

---

Fig. 1. Pseudo code for algorithm \( \text{OptPathTrans} \).
Fig. 2. Sampling time instants for Mobile Path $P_S$ (determined by Algorithm $OptPathTrans$).

Fig. 2. Sampling time instants for Mobile Path $P_S$ (hypothesis for the proof).

2.1. Sampling time instants for Mobile Path $P_S$ (determined by Algorithm $OptPathTrans$).

2.2. Sampling time instants for Mobile Path $P_S$ (hypothesis for the proof).

2.6. Example run of algorithm $OptPathTrans$

Consider the mobile graph $G_M = G_1G_2G_3G_4G_5$ (Fig. 3.1), 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.

The execution of algorithm $OptPathTrans$ on the mobile graph $G_M$ of Fig. 3.1 is shown in Fig. 3.2. 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.

### 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\textsuperscript{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 assumes that each node continues to move in its current direction of motion with the current velocity. Given the network dimensions $(0 \leq X_{\text{max}}, 0 \leq Y_{\text{max}})$, the location $(x_i, y_i)$ of a node $i$ at time instant $t$, the direction of motion $\theta$ $(0 \leq \theta \leq 360)$ with reference to the positive $x$-axis, and the current velocity $v_i$, 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:

$$
x_i^{t+\delta t} = \begin{cases} 
    x_i' + (v_i' \cdot \delta t \cdot \cos \theta) & \text{if } 0 \leq \theta \leq 90 \\
    x_i' - (v_i' \cdot \delta t \cdot \cos(180 - \theta)) & \text{if } 90 \leq \theta \leq 180 \\
    x_i' - (v_i' \cdot \delta t \cdot \cos(\theta - 180)) & \text{if } 180 \leq \theta \leq 270 \\
    x_i' + (v_i' \cdot \delta t \cdot \cos(360 - \theta)) & \text{if } 270 \leq \theta \leq 360 
\end{cases}
$$

$$
y_i^{t+\delta t} = \begin{cases} 
    y_i' + (v_i' \cdot \delta t \cdot \sin \theta) & \text{if } 0 \leq \theta \leq 90 \\
    y_i' + (v_i' \cdot \delta t \cdot \sin(180 - \theta)) & \text{if } 90 \leq \theta \leq 180 \\
    y_i' - (v_i' \cdot \delta t \cdot \sin(\theta - 180)) & \text{if } 180 \leq \theta \leq 270 \\
    y_i' - (v_i' \cdot \delta t \cdot \sin(360 - \theta)) & \text{if } 270 \leq \theta \leq 360 
\end{cases}
$$

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

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

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

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 $G_{\text{pred}}^{t}(t,t+\delta t) = G_t \cap G_{t+1}^{\text{pred}} \cap \cdots \cap G_{t+\delta t}^{\text{pred}}$. If a minimum hop $s-d$ path exists in $G_{\text{pred}}^{t}(t,t+\delta t)$, then that path is validated in the actual mobile sub graph $G_{\text{actual}}^{t}(t,t+\delta t) = G_t \cap G_{t+1} \cap G_{t+2} \cap \cdots \cap G_{t+\delta t}$ that spans time instants $t, t+1, t+2, \ldots, t+\delta t$. If an $s-d$ path exists in both $G_{\text{pred}}^{t}(t,t+\delta t)$ and $G_{\text{actual}}^{t}(t,t+\delta t)$, then that $s-d$ path is used at time instants $t, t+1, \ldots, t+\delta t$.

If an $s-d$ path exists in $G_{\text{pred}}^{t}(t,t+\delta t)$, $G_{\text{pred}}^{t}(t,t+\delta t+1)$ and $G_{\text{actual}}^{t}(t,t+\delta t)$, but not in $G_{\text{actual}}^{t}(t,t+\delta t+1)$, 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 $G_{\text{pred}}^{t}(t,t+\delta t)$ and $G_{\text{actual}}^{t}(t,t+\delta t)$, but not in $G_{\text{pred}}^{t}(t,t+\delta t+1)$, 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.

2.8. Look-ahead window

In this approach, we use the notion of a look-ahead window size, $\Delta$, as the time for which accurate information about the future topology changes are known. In other words, if a source node has to choose the longest living path since time instant $t$, the source node does not know the future topology changes from $t$ to the end of the simulation, but knows only the network topology changes from $t$ to $t+\Delta$, where $\Delta$ is the look-ahead window size whose value can range from 0 to the duration of the $s-d$ session, $T$.

When an $s-d$ path is required at time instant $t$, we try to find the minimum hop $s-d$ path in the mobile sub graph $G(t,t+\Delta) = G_t \cap G_{t+1} \cap \cdots \cap G_{t+\Delta}$. If a minimum hop path exists in $G(t,t+\Delta)$, we use that path from $t$ to $t+\Delta$ and continue to use the path beyond $t+\Delta$ as long as the path exists. If a path does not exist in $G(t,t+\Delta)$, we find a mobile sub graph $G(t,t+\delta)$ such that $\delta < \Delta$ and there exists at least one $s-d$ path in $G(t,t+\delta)$ and no $s-d$ path exists in $G(t,t+\delta+1)$. We then select the minimum hop $s-d$ path in $G(t,t+\delta)$. If $t+\delta < T$, we repeat the above procedure by setting $t = t+\delta+1$. The sequence of paths obtained under this approach of look-ahead window will be denoted as Stable-Mobile-Path$^\text{Look-ahead}$.

2.9. Simulations

We use the Random Waypoint mobility model [3], one of the most widely used models for simulating mobility in 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 $[v_{\text{min}} \cdots v_{\text{max}}]$. 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 $v_{\text{min}} = 0$ and pause time of a node is 0. The values of $v_{\text{max}}$ 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). In future, we will evaluate the performance of our algorithms with other MANET mobility models like Random Walk, Random Direction and the Gauss-Markov models [6].

2.9.1. Simulation conditions

We ran our simulations with a square topology of dimensions $1000 \times 1000$ m. The wireless transmission range of a node is 250 m. The node density is varied by performing the simulations in this network with 50 (10 neighbors per node), 100 (20 neighbors per node) and 150 nodes (30 neighbors per node). Note that, two nodes $a$, $b$ 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 s in the ns-2 network simulator [9]. Each data point in Figs. 4–6 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 and 20 s. The simulation conditions are summarized in Table 1.

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 s, respectively, then the time-averaged

Fig. 5. Performance of Stable Mobile Path Look-ahead with $[v_{min} \cdot v_{max}] = [0 \cdots 10 \text{ m/s}]$.

5.1. Stability of routes (50 nodes).

5.2. Stability of routes (150 nodes).

5.3. Hop count of routes (50 nodes).

5.4. Hop count of routes (150 nodes).
The hop count of the mobile path would be \((2 \times 2 + 3 \times 3 + 2 \times 5)/10 = 2.3\).

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 end of the simulation time.

We study the performance of Minimum-Hop-Mobile-Path, Stable-Mobile-Path and Stable-Mobile-Path\(^{Uncertain-Pred}\) under each of the node density, mobility and network size conditions listed in Table 1. For lack of space, we only show simulation results for the Stable-Mobile-Path\(^{Look-ahead}\) obtained under extreme cases of node mobility (i.e., \(v_{\text{max}} = 10\) and \(50\) m/s) and node density (50 nodes and 150 nodes). We vary the look-ahead window size \(\Delta\) from 0 to \(\Delta_{\text{max}}\), where \(\Delta_{\text{max}}\) is the look-ahead window size beyond which there is no impact on the number of route transitions or the hop count. In other words, \(\Delta_{\text{max}}\) is an upper bound on the lifetime of any \(s-d\) path and depends on the node density and velocity.

### 2.9.2. 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 5–7 times to that of the optimal number of route transitions for a low-density network (refer Fig. 4.1) and 8–10 times to that of the optimal for a high-density network (refer Fig. 4.3). The average hop count per Stable-Mobile-Path is 1.5–1.8 times to that of the optimal hop count incurred in a low-density network (refer Fig. 4.2) and is 1.8 to 2.1 times to that of the optimal hop count incurred in a high-density network (refer Fig. 4.3).
of the optimal in a high-density network (refer Fig. 4.4). Optimality in both these metrics cannot be obtained simultaneously.

### 2.9.3. 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. 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 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. 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.

### 2.9.4. Impact of node density

When we aim for minimum hop $s$–$d$ paths, we target paths that can go through the minimum number of intermediate nodes between the source $s$ and destination $d$. As we increase the node density, there are more neighbors per node and thus increasing the probability of finding a far-away neighbor. 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 $r_{max}$, minimum hop paths are more stable in low-density networks compared to high-density networks (compare Figs. 4.1 and 4.3), while 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 Figs. 4.2 and 4.4).

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 at high-density networks compared to low-density networks.

### 2.9.5. Performance of Stable-Mobile-PathUncertain-pred

The number of route transitions incurred by Stable-Mobile-PathUncertain-pred is only at most 1.6–1.8 times that of the optimal for low-density networks (refer Fig. 4.1) and 2–3 times that of optimal for high-density networks (refer Fig. 4.3). Nevertheless, the average hop count incurred by Stable-Mobile-PathUncertain-pred is 1.3–1.6 times to that incurred by Minimum-Hop-Mobile-Path (refer Figs. 4.2 and 4.4).

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 as part of the on-demand route discovery process. Rather than just arbitrarily choosing a minimum hop path that the RREQ packet has traversed through and sending a route reply (RREP) packet along that path, the destination node can construct a mobile sub graph that will incorporate the locations of nodes in the near future, apply algorithm OptPathTrans, obtain Stable-Mobile-PathUncertain-pred and send the RREP along that path. We are currently developing the above approach into an on-demand stable path routing protocol. We will also evaluate the performance of algorithm OptPathTrans when the future locations of nodes are predicted with other uncertainty models.

### 2.9.6. Performance of Stable-Mobile-PathLook-ahead

When the look-ahead window size $A$ is 0, we obtain the performance of a Minimum-Hop-Mobile-Path. As we increase $A$, we run algorithm OptPathTrans on an intersection of static graphs spanned by a time period equal to $A$. We only consider edges that exist at least for the period of $A$ to be part of the paths. For smaller values of $A$, we will
have more edges that exist for at least $A$ seconds. As we increase $A$, the number of common edges that exist for a time period equal to $A$ starts decreasing significantly.

We observe a critical value for the look-ahead window size, $A_{\text{critical}}$, for each of the simulation conditions. For the look-ahead window size values up to $A_{\text{critical}}$, the rate of decrease in the number of route transitions is high while there is a minimum increase in the average hop count per path. Such a contrasting tradeoff between stability and number of edges can be attributed to the presence of multiple common edges when looked at the near future (i.e., at least for the next $A_{\text{critical}}$ seconds) and the capability of the algorithm to select only among them. For look-ahead window size values from $A_{\text{critical}}$ up to $A_{\text{max}}$, the rate of decrease in the number of route transitions with respect to $A$ is relatively slow, while the rate of increase in the average hop count per path is relatively high, compared to that observed when $A \leq A_{\text{critical}}$. This could be attributed to the predominant presence of links whose average physical distance is only 50–60% of the transmission range of the nodes. But, to span the path from the source node to the destination node, more such links have to be accommodated.

For example, in low-mobility scenarios (i.e., $v_{\text{max}} = 10$ m/s, refer Fig. 5), the number of path transitions decreases by a factor of 2.5–3 when $A$ is increased from 0 to 20 s. The corresponding increase in the average hop count per path is only by a factor of 15%. When $A$ is increased from 20 s to $A_{\text{max}}$, which is like 175 s, the relative decrease (i.e., compared to the route transitions at $A_{\text{critical}}$) in the number of route transitions is by a factor of 3. At the same time, the relative increase in the average hop count per path is by a factor of 60–100%. Similar observations can be made for high-mobility scenarios (refer Fig. 6), where $A_{\text{critical}}$ is observed to be 10 s and $A_{\text{max}}$ is 80–100 s. The value of $A_{\text{critical}}$ and $A_{\text{max}}$ decreases with increase in the node mobility. This is obvious because, as the topology changes dynamically, there is no need to look ahead in the far future.

3. Multicast Steiner tree

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.

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 paper, we assume the weight of each edge is unity 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 (i.e., 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., [13]) have been proposed in the literature to approximate a minimum Steiner tree.

We show that 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 [18]. 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 [13] well-
known \( O(|V||S|^2) \) heuristic, as the underlying heuristic to determine the longest existing multicast tree. We also define the Minimum-Mobile-Multicast-Steiner-Tree as the sequence of approximations to the minimum Steiner tree obtained by directly using Kou’s heuristic whenever required.

### 3.1. Heuristic to approximate minimum steiner tree

We use the Kou et al.’s [13] 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. 7. 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 \(s \in S\).

### 3.2. Algorithm OptTreeTrans

Let \(G_M = G_1G_2 \cdots G_T\) 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 \(G(i,j) = G_i \cap G_{i+1} \cap \cdots \cap G_j\) 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. 8.

### 3.3. Algorithm complexity and proof of correctness

In a mobile graph \(G_M = G_1G_2 \cdots G_T\), 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 \(Opt\) is \(O(|V||S|^2 T)\).

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

Let \(epoch^1_S, epoch^2_S, \ldots, epoch^m_S\) be the set of sampling time instants in each of which the Mobile-Multicast-Steiner-Tree \((s,S)\)MobileStabletree suffers no tree transitions. Let \(epoch^1'_S, epoch^2'_S, \ldots, epoch^m'_S\) be the set of sampling time instants in each of which the Mobile-Multicast-Steiner-Tree \((s,S)\)MobileStabletree suffers no tree transitions. Let \(t_{init}^S, t_{end}^S\) be the initial and final sampling time instants of \(epoch^k_S\) where \(1 \leq k \leq m\). Let \(t_{init}^S, t_{end}^m\) be the initial and final sampling time instants of \(epoch^k'_S\) where \(1 \leq k \leq m'\). Note that \(t_{end}^S = t_{init}^m\) and \(t_{end}^m = t_{init}^m\) to indicate

\[
\begin{array}{l}
\text{Input:} \quad \text{An undirected graph } G = (V, E) \\
\text{Multicast group } S \subseteq V \\
\text{Output:} \quad \text{A tree } T_M \text{ for the set } S \text{ in } G \\
\end{array}
\]

**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 \(T_M\), from \(T_S\) by deleting edges in \(T_S\), if necessary, such that all the leaves in \(T_M\) are members of \(S\).

---

Fig. 7. Kou et al.’s Heuristic [13] to find an approximate Minimum Steiner Tree.
3.4. Example run of algorithm OptTreeTrans

Consider the mobile graph \( G_M = G_1G_2G_3G_4G_5 \) sampled every second (Fig. 9.1). 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 would be 5 and the time averaged number of edges per Minimum-Mobile-Steiner-Tree would be 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. 9.2. 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\} \( G_{125} \)\}. The number of tree transitions is 2 and the time-averaged number of edges in the Stable-Mobile-Steiner-Tree is \((4 \times 2 + 5 \times 3)/5 = 4.6\) 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 \( G_5 \). Such a tradeoff between the number of Steiner tree transitions and number of edges in the mobile Steiner tree is also vindicated by simulation results.

3.5. Simulations

The simulation conditions are summarized in Table 2. Random waypoint mobility model is the mobility model used in all the simulations. Each multicast group includes a source node and a set.
of receiver nodes. Each data point in Figs. 10–19 is an average of simulations conducted over 10 mobility trace files and 5 randomly selected groups of a particular size. The origination time for a multicast session is uniformly distributed in the range 1–50 s. Once begun, a multicast session lasts until the end of the simulation.

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 number of links in the constituent \((s-S)\) Steiner trees, averaged over time.

We study the performance of Minimum-Mobile-Multicast-Steiner-Tree, Stable-Mobile-Multicast-Steiner-Tree, Stable-Mobile-Multicast-Steiner-Tree\(^{\text{Uncertain-pred}}\) (the sequence of stable multicast Steiner trees obtained when algorithm \(\text{OptTreeTrans}\) is run on the static graphs obtained using the prediction with uncertainty approach) under each of the network density, node mobility and multicast group size conditions listed in Table 2. For lack of space, we only show simulation results for the Stable-Mobile-Multicast-Steiner-Tree\(^{\text{Look-ahead}}\) (the sequence of stable paths obtained when algorithm \(\text{OptTreeTrans}\) is run on the static graphs generated using look-ahead window size approach) obtained under multicast group sizes of 4 and 24. We vary the look-ahead window size \(A\) from 0 to \(A_{\text{max}}\), where \(A_{\text{max}}\) is the look-ahead window size beyond which there is no impact on the number of tree transitions or the number of edges.

Table 2
Simulation conditions for Algorithm \(\text{OptTreeTrans}\)

<table>
<thead>
<tr>
<th>Network size</th>
<th>1000 m (\times) 1000 m</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of nodes</td>
<td>50 and 150</td>
</tr>
<tr>
<td>Simulation time</td>
<td>1000 s</td>
</tr>
<tr>
<td>Transmission range</td>
<td>250 m</td>
</tr>
<tr>
<td>Multicast group size</td>
<td>2, 4, 8, 12, 18 and 24</td>
</tr>
<tr>
<td>Topology sampling interval</td>
<td>1 s</td>
</tr>
<tr>
<td>Multicast routing algorithms</td>
<td>Kou’s minimum Steiner Tree heuristics and Algorithm (\text{OptTreeTrans}) (run using complete knowledge of future topology changes, prediction with uncertainty, look-ahead window size)</td>
</tr>
<tr>
<td>Minimum node speed</td>
<td>0 m/s</td>
</tr>
<tr>
<td>Maximum node speed</td>
<td>10 and 50 m/s</td>
</tr>
<tr>
<td>Pause time</td>
<td>0 s</td>
</tr>
</tbody>
</table>
Fig. 10. Performance of Minimum-Mobile-Steiner-Tree, Stable-Mobile-Steiner-Tree, Stable-Mobile-Steiner-Tree\textsuperscript{Uncertain-pred} with $[v_{\text{min}}, v_{\text{max}}] = [0 \cdot 10 \text{ m/s}]$ (network of 50 nodes).

Fig. 11. Performance of Minimum-Mobile-Steiner-Tree, Stable-Mobile-Steiner-Tree, Stable-Mobile-Steiner-Tree\textsuperscript{Uncertain-pred} with $[v_{\text{min}}, v_{\text{max}}] = [0 \cdot 10 \text{ m/s}]$ (network of 150 nodes).

Fig. 12. Performance of Minimum-Mobile-Steiner-Tree, Stable-Mobile-Steiner-Tree, Stable-Mobile-Steiner-Tree\textsuperscript{Uncertain-pred} with $[v_{\text{min}}, v_{\text{max}}] = [0 \cdot 50 \text{ m/s}]$ (network of 50 nodes).

Fig. 13. Performance of Minimum-Mobile-Steiner-Tree, Stable-Mobile-Steiner-Tree, Stable-Mobile-Steiner-Tree\textsuperscript{Uncertain-pred} with $[v_{\text{min}}, v_{\text{max}}] = [0 \cdot 50 \text{ m/s}]$ (network of 150 nodes).

Fig. 14. Stability of Stable-Mobile-Multicast-Steiner-Tree\textsuperscript{Look-ahead}, $v_{\text{max}} = 10 \text{ m/s}$ (multicast group size – 4 nodes).
Fig. 15. Stability of Stable-Mobile-Multicast-Steiner-Tree\textsuperscript{look-ahead}, $v_{\text{max}} = 10$ m/s (multicast group size – 24 nodes).

Fig. 16. Stability of Stable-Mobile-Multicast-Steiner-Tree\textsuperscript{look-ahead}, $v_{\text{max}} = 50$ m/s (multicast group size – 4 nodes).

Fig. 17. Stability of Stable-Mobile-Multicast-Steiner-Tree\textsuperscript{look-ahead}, $v_{\text{max}} = 50$ m/s (multicast group size – 24 nodes).

Fig. 18. Edges per Stable-Mobile-Multicast-Steiner-Tree\textsuperscript{look-ahead} (multicast group size – 4 nodes).

Fig. 19. Edges per Stable-Mobile-Multicast-Steiner-Tree\textsuperscript{look-ahead} (multicast group size – 24 nodes).
3.5.1. Minimum-Mobile-Multicast-Steiner-Tree vs. Stable-Mobile-Multicast-Steiner-Tree

Stability of Minimum-Mobile-Multicast-Steiner-Trees decreases rapidly with increase in multicast group size (refer Figs. 10–13). On the other hand, by accommodating 10–40% more edges (refer Figs. 10–13), stability of the Stable-Mobile-Multicast-Steiner-Tree is almost insensitive to multicast group size. For given value of \( v_{\text{max}} \), the number of tree transitions incurred by the Minimum-Mobile-Multicast-Steiner-Tree in a low-density network (refer Figs. 10 and 12) 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 Figs. 11 and 13), 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 \( \text{OptTreeTrans} \) makes use of the available nodes and links as much as possible in order to maximize the stability of the trees. On the contrary, in the case of a Minimum-Mobile-Steiner-Tree, the average number of links in the constituent (S-S) trees remains the same with increase in node density; the stability of the Minimum-Mobile-Steiner-Trees decreases with increase in node density.

For a given value of \( v_{\text{max}} \), the number of tree transitions incurred by the Stable-Mobile-Multicast-Steiner-Tree in uncertain-pred in a low-density network (refer Figs. 10 and 12) 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 Figs. 11 and 13), 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 in uncertain-pred is not much affected by multicast group size and the number of edges (refer Figs. 10–13) is at most 40% more than that found in a Minimum-Mobile-Multicast-Steiner-Tree.

3.5.2. Performance of Stable-Mobile-Multicast-Steiner-Tree \( ^{\text{Look-ahead}} \)

When the look-ahead window size, \( \Delta \), is 0, we obtain the performance of the Minimum-Mobile-Steiner-Tree. As \( \Delta \) is increased, algorithm \( \text{OptTreeTrans} \) becomes stability conscious. In low-mobility scenarios, the value of \( \Delta_{\text{critical}} \) is 10 (Fig. 15, with group size of 24) to 20 s (Fig. 14, with group size of 4). As observed in these figures, the number of tree transitions decreases by a factor of 3 (with a group size of 4) to 5 (with a group size of 24) by looking ahead for \( \Delta_{\text{critical}} \) seconds. At the same time, the increase in the number of edges is only by a factor of 3–10% compared to that of a Minimum-Mobile-Steiner-Tree (refer Figs. 18 and 19). In high-mobility scenarios, the value of \( \Delta_{\text{critical}} \) is 5 (Fig. 17, with group size of 24) to 10 s (Fig. 16, with group size of 4). As observed in these figures, the number of tree transitions decreases by a factor of 4 (with a group size of 4) to 7 (with a group size of 24) by looking ahead for \( \Delta_{\text{critical}} \) seconds. At the same time, the increase in the number of edges is only by a factor of 5–10% (refer Figs. 18 and 19). This shows that multicast group members can be kept connected for a relatively long time by accommodating few more edges than that required by a Minimum-Mobile-Steiner-Tree. Such a contrasting tradeoff between stability and number of edges can be attributed to the presence of multiple common edges when looked at the near future, and the capability of the algorithm \( \text{OptTreeTrans} \) to select only those edges.

In low-density networks (refer Figs. 14–17), the relative decrease in the number of tree transitions at \( \Delta_{\text{max}} \) compared to those incurred at \( \Delta_{\text{critical}} \) is by a factor of 1.7 (at high-mobility scenarios) to 2.5 (at low-mobility scenarios), accompanied by an increase in the number of constituent edges of the Steiner tree (refer Figs. 18 and 19) by a factor of 10% (for group size of 24)–40% (for group size of 4). Similarly, in high-density networks (refer Figs. 14–17), the decrease in the number of tree transitions at \( \Delta_{\text{max}} \) compared to those incurred at \( \Delta_{\text{critical}} \) is by a factor of 2.5 (at high-mobility scenarios) to 5 (at low-mobility scenarios), accompanied by an increase in the number of constituent edges (refer Figs. 18 and 19) by a factor of 25% (for group size of 24)–60% (for group size of 4). Thus, increasing the network density helps to improve the stability of the trees, with a marginal increase in the number of constituent edges.

4. Connected dominating set

Given a connected undirected graph, \( G = (V, E) \) where \( V \) is the set of vertices and \( E \) is the set of edges, a connected dominating set (CDS) is a subgraph of \( G \) such that all nodes in the graph are included in the CDS or directly attached to a node in the CDS. A minimum connected dominating set
(MCDS) is the smallest CDS (in terms of number of nodes in the CDS) for the entire graph.

Route discovery in ad hoc networks is often source-initiated and is often accomplished using a broadcast request and reply cycle. As ad hoc networks are often multi-hop in nature, the request packets need to be forwarded by more than one node in order to make sure that all nodes are considered for route selection. The set of nodes that forward the route request packets is said to form a CDS. If all nodes in the network forward the route request packet once (a situation called flooding), each node will receive the same request packet from multiple nodes. The smaller the size of the CDS, lower the number of unnecessary retransmissions. If the route request packet is forwarded by only the nodes in the MCDS, we will have the minimum number of retransmissions. Unfortunately, the problem of determining the MCDS in an undirected graph like that of the unit disk graph is NP-complete. Efficient heuristics $[2,4,5]$ have been proposed to approximate the MCDS in wireless ad hoc networks.

We call the sequence of approximations of a MCDS over a given mobile graph, as the Minimum-Mobile-connected-dominating-set (Minimum-Mobile-CDS). We observe that the Minimum-Mobile-CDS is very unstable as it has minimal nodes and links and hence incurs many transitions $[18]$. Either the MCDS might become disconnected or even if the MCDS stays connected, it may not be covering certain nodes in the network. In either case, the MCDS requires a reconstruction and frequent CDS reconstructions will add considerable overhead to the already resource-constrained ad hoc networks. We show that by adding a few more nodes to the CDS, we can improve its stability. Similar to our approach to find stable paths and trees, we propose an efficient, polynomial-time greedy algorithm called OptCDSTrans, to determine the minimum number of CDS transitions during the time period of a network session in which one or more source–destination pairs communicate. Given the complete knowledge of future topology changes, the algorithm operates on the following principle: Whenever a CDS is required, choose the CDS that will exist for the longest time. The above strategy is repeated over the duration of the network session and the sequence of stable connected dominating sets obtained by running this algorithm is called the Stable-Mobile-connected-dominating-set (Stable-Mobile-CDS).

4.1. Minimum connected dominating set

We could not find a heuristic that will include a specific node (the source node) in the MCDS such that the route-request packets originating at the source node can be broadcast throughout the network with the minimum number of retransmissions. In this direction, we propose a heuristic called $d$-MCDS that makes use of the well-known greedy degree-based approach of iteratively including the nodes (into the CDS) with the largest number of uncovered neighbors (i.e., nodes that are not already in the CDS or attached to a node in the CDS). We use $d$-MCDS (pseudo code in Fig. 20) as the underlying algorithm to determine the constituent connected dominating sets of the Stable-Mobile-CDS.

The idea of $d$-MCDS is to start with a source node (included in the CDS) and increase the CDS size iteratively including only the node that has the maximum number of uncovered neighbors. The above procedure is repeated until all nodes in the network graph are covered by the CDS. In each of the iterations, exactly one node will be added to the CDS and there would be at most $|V|$ iterations, where $|V|$ is the number of vertices in the graph. The vertex with the maximum number of uncovered neighbors can be determined in $O(|E|)$ time by scanning the adjacency list of the graph, where $|E|$ is the number of edges in the graph. At the worst, $|E| = |V|^2$. Thus, the overall complexity of $d$-MCDS is $O(|V|^3)$.

We compare the performance of $d$-MCDS (in terms of the number of constituent nodes in the approximate MCDS) with other heuristics proposed (AWF $[2]$, BCDP $[4]$ and BCOP $[5]$) for determining an approximate MCDS in the context of wireless ad hoc networks. We do not implement these heuristics; we take their performance results presented in $[5]$. The results are presented in Table 3. Each entry in the last column of Table 3 is an average of the $d$-MCDS heuristic run on 30 instances of randomly distributed vertices generated for each of the planar areas listed in the first column of the table, with the source node randomly selected in each case. The performance of $d$-MCDS vis-à-vis the centralized version of BCOP shows $d$-MCDS as a better heuristic when the average degree of a node is less than 15. For larger node degrees, $d$-MCDS performs only slightly worse than the centralized version of BCOP.
4.2. Algorithm OptCDSTrans

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 network session. When a CDS 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 CDS in $G(i, j)$ and no CDS exists in $G(i, j+1)$ or $j = T$. A connected dominating set in $G(i, j)$ is selected using heuristic $d$-MCDS. Such a CDS exists in each of the static graphs $G_i, G_{i+1}, \ldots, G_j$. If there is a tie, the CDS with the smallest number of constituent nodes is chosen. If $t_{j+1} \leq t_T$, the above procedure is repeated by finding the CDS that can survive for the maximum amount of time since $t_{j+1}$. A sequence of such maximum lifetime connected dominating sets over the timescale of $G_M$ forms the Stable-Mobile-connected-dominating-set. The pseudo code is given in Fig. 21.

4.3. Algorithm complexity and proof of correctness

Given a mobile graph $G_M = G_1 G_2 \ldots G_T$, spanning $T$ static graphs, there can be at most $T$ transitions. Heuristic $d$-MCDS would have to be thus run
at most $T$ times, each time on a graph of $|V|$ nodes. As the worst-case run-time complexity of $d$-MCDS is $O(|V|^3)$, the worst-case run-time complexity of algorithm $OptCDSTrans$ is $O(|V|^3T)$.

Given a mobile graph $G_M = G_1G_2 \cdots G_T$, let the number of CDS transitions in the Mobile-CDS $CDS_S$ generated by $OptCDSTrans$ be $m$. To prove $m$ is the optimal number of CDS transitions, assume the contrary: there exists a Stable-Mobile-CDS $CDS_S$ in $G_M$ and the number of CDS transitions in $CDS_S$ is $m' < m$.

Let $epoch_1, epoch_2, \ldots, epoch_m$ be the set of sampling time instants in each of which the mobile connected dominating set $CDS_S$ suffers no CDS transitions. Let $epoch_1', epoch_2', \ldots, epoch_m'$ be the initial and final sampling time instants of $epoch_j$ where $1 \leq j \leq m$. Let $t_{S_j}^{\text{init}}, t_{S_j}^{\text{end}}$ be the initial and final sampling time instants of $epoch_j' | S_j$ where $1 \leq k \leq m'$. Note that $t_{S_j}^{\text{init}}$, $t_{S_j}^{\text{end}}$, $t_{S_j}^{\text{init}}$, $t_{S_j}^{\text{end}}$, $t_{S_j}^{k}$, and $t_{S_j}^{end}$ must indicate $CDS_S$ and $CDS_S$ span over the same time period, $T$, of the network session.

Now, since we claim that $m' < m$, there should exist $j$, $k$ where $1 \leq j \leq m$ and $1 \leq k \leq m'$ such that $epoch_j \subseteq epoch_k'$, i.e., $t_{S_j}^{\text{init}}, t_{S_j}^{\text{end}} < t_{S_k}^{\text{init}}, t_{S_k}^{\text{end}} < t_{S_j}^{\text{k}}, t_{S_j}^{end}$ and at least one CDS existed in $[t_{S_j}^{\text{k}}, t_{S_j}^{end}]$. In other words, there should be at least one CDS in $CDS_S$ that has a lifetime larger than that of the life-time of any CDS in $CDS_S$. But, algorithm $OptCDSTrans$ made a route transition at $t_{S_j}^{end}$ since there was no CDS from $t_{S_j}^{\text{init}}$ beyond $t_{S_j}^{\text{end}}$. Thus, there is no CDS in the range $[t_{S_j}^{\text{k}}, t_{S_j}^{end}]$ and hence there is no CDS in the range $[t_{S_j}^{\text{k}}, t_{S_j}^{end}]$. This shows that the lifetime of each of the CDS in $CDS_S$ has to be smaller or equal to the lifetime of the CDS in $CDS_S$, implying $m' \geq m$. This is a contradiction and proves that our hypothesis $m' < m$ is not correct. Hence, the number of CDS transitions in $CDS_S$ is optimal and $CDS_S$ is the Stable-Mobile-CDS.

4.4. Example run of algorithm $OptCDSTrans$

Consider the mobile graph $G_M = G_1G_2G_3G_4G_5$, (Fig. 22.1), generated with a sampling interval of one second. The source node of the broadcasting process is node 4. The sequence of minimum CDS in $G_M$ would be $\{4–3, G_12\}, \{4–2, G_4, 4–2, 2–3\}G_4, \{4–2\}G_3$. The average size of the Minimum-Mobile-CDS with respect to time would be $(2 \times 2 + 2 \times 1 + 3 \times 1 + 2 \times 1)/5 = 2.2$ and the number of CDS transitions is 4. When we aim for the sequence of CDS that will exist for the longest amount of time in the mobile graph, we apply algorithm $OptCDSTrans$ and get the following stable mobile CDS: $\{4–5, 5–6, 6–3\}G_{13}, \{4–2, 2–5\}G_{45}$. The time-averaged size of the Stable-

<table>
<thead>
<tr>
<th>Size</th>
<th>Radius (m)</th>
<th>Average degree</th>
<th>Number of nodes in the approximate MCDS</th>
<th>AWF</th>
<th>BCDP</th>
<th>BCOP (distributed)</th>
<th>BCOP (centralized)</th>
<th>d-MCDS (centralized)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 m × 100 m</td>
<td>20</td>
<td>10.74</td>
<td>28.21</td>
<td>20.11</td>
<td>20.68</td>
<td>19.18</td>
<td>18.68</td>
<td></td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>15.45</td>
<td>20.00</td>
<td>13.80</td>
<td>14.30</td>
<td>12.67</td>
<td>12.67</td>
<td></td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>26.24</td>
<td>15.07</td>
<td>10.07</td>
<td>10.17</td>
<td>9.00</td>
<td>8.93</td>
<td></td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>27.49</td>
<td>11.67</td>
<td>7.73</td>
<td>8.17</td>
<td>6.30</td>
<td>6.83</td>
<td></td>
</tr>
<tr>
<td>120 m × 120 m</td>
<td>25</td>
<td>11.21</td>
<td>26.56</td>
<td>18.26</td>
<td>19.00</td>
<td>17.78</td>
<td>17.52</td>
<td></td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>15.82</td>
<td>20.67</td>
<td>13.40</td>
<td>14.63</td>
<td>12.40</td>
<td>12.33</td>
<td></td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>20.53</td>
<td>15.87</td>
<td>10.03</td>
<td>11.27</td>
<td>9.13</td>
<td>9.47</td>
<td></td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>25.80</td>
<td>12.67</td>
<td>8.33</td>
<td>9.00</td>
<td>7.00</td>
<td>7.57</td>
<td></td>
</tr>
<tr>
<td>140 m × 140 m</td>
<td>30</td>
<td>12.46</td>
<td>25.76</td>
<td>18.10</td>
<td>18.90</td>
<td>17.03</td>
<td>16.54</td>
<td></td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>15.74</td>
<td>20.00</td>
<td>13.33</td>
<td>14.37</td>
<td>12.73</td>
<td>12.51</td>
<td></td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>19.15</td>
<td>16.40</td>
<td>10.87</td>
<td>11.57</td>
<td>9.67</td>
<td>9.90</td>
<td></td>
</tr>
<tr>
<td></td>
<td>45</td>
<td>23.38</td>
<td>13.33</td>
<td>9.03</td>
<td>9.40</td>
<td>7.77</td>
<td>8.07</td>
<td></td>
</tr>
<tr>
<td>160 m × 160 m</td>
<td>30</td>
<td>9.12</td>
<td>31.88</td>
<td>22.20</td>
<td>23.20</td>
<td>21.88</td>
<td>21.07</td>
<td></td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>12.17</td>
<td>24.50</td>
<td>17.07</td>
<td>17.36</td>
<td>16.18</td>
<td>16.10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>15.41</td>
<td>19.73</td>
<td>13.47</td>
<td>14.07</td>
<td>12.43</td>
<td>12.57</td>
<td></td>
</tr>
<tr>
<td></td>
<td>45</td>
<td>18.95</td>
<td>16.33</td>
<td>11.07</td>
<td>11.40</td>
<td>9.93</td>
<td>10.03</td>
<td></td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>22.65</td>
<td>13.80</td>
<td>9.47</td>
<td>9.67</td>
<td>7.93</td>
<td>8.63</td>
<td></td>
</tr>
</tbody>
</table>
Mobile-CDS would be \((4 \times 3 + 3 \times 2)/5 = 3.6\), while the number of CDS transitions is 2. We also observe such a tradeoff between the CDS size and the number of CDS transitions in the simulations.

4.5. Simulations

We use the Random Waypoint mobility model as the mobility model for all our simulations. The starting time of each of the simulations is the first second. Each data point in Figs. 23–26 is an average computed over 10 mobility trace files and 8 randomly selected source nodes. The simulation conditions are summarized in Table 4. The performance metrics evaluated are the number of CDS transitions and the average CDS size in the mobile connected dominating sets, which is the number of nodes in the constituent connected dominating sets, averaged over time.

When a MCDS is required at sampling time instant \(t_i\) and stability is not to be considered, we run the \(d\)-MCDS heuristic on the static graph \(G_t\) and the approximate MCDS 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 connected dominating sets generated by the above strategy as the Minimum-Mobile-CDS. We study the performance of Minimum-Mobile-CDS, Stable-Mobile-CDS, Stable-Mobile-CDS\(^{Uncertain\text{-}pred}\) (sequence of connected dominating sets obtained when algorithm OptCDSTrans is run on the static graphs obtained using the prediction with uncertainty approach) under each of the network density and node mobility conditions listed in Table 4. For lack of space, we only show simulations results for the Stable-Mobile-CDS\(^{Look\text{-}ahead}\) (sequence of connected dominating sets obtained when algorithm OptCDSTrans is run on static graphs generated using the look-ahead window approach) obtained under extreme cases of node mobility (i.e., \(v_{\text{max}} = 10\) and 50 m/s) and network density (50 and 150 nodes).

4.5.1. Minimum-Mobile-CDS vs. Stable-Mobile-CDS

MCDS includes only the minimum set of nodes connected among themselves and each other node in the network is directly linked to at least one node in the MCDS. Such a CDS needs to comprise links that will have to span over long distances in order to cover the whole network with minimum number of constituent nodes. Thus, probability of a MCDS getting disconnected in the near future is high. Also, as nodes move dynamically, chances of a non-MCDS node always being in the vicinity of a node belonging to the MCDS is less. Performance results

\[\text{Fig. 22. Minimum-Mobile-Connected-Dominating-Set vs. Stable-Mobile-Connected-Dominating-Set.}\]
(refer Fig. 23) show as we accommodate 30% (low density)–100% (high density) more nodes in the CDS, the number of CDS transitions in the Stable-Mobile-CDS is 5 (low density) to 13 (high density) times less than that incurred in Minimum-Mobile-CDS.

The number of nodes in MCDS that form part of the Minimum-Mobile-CDS is 13 (network of 50 nodes).
nodes) and 17 (network of 150 nodes). With increase in the network density, the number of transitions incurred by Minimum-Mobile-CDS increases by a factor of 60 to 140%, apparently to cover more nodes in a dynamically changing network. On the other hand, the average number of transitions of Stable-Mobile-CDS at high network density is 15–20% less than that observed at low network density; while the average CDS size of a stable mobile CDS at high network density is 100% more than that observed at low network density. For a given value of network density, the number of CDS transitions incurred in Stable-Mobile-CDS Uncertain-Pred is only at most 1.75 times to that of the optimal number of CDS transitions. At the same time, the average number of nodes in the constituent CDS of the Stable-Mobile-CDS Uncertain-Pred is only 30–60% more than that incurred in a Minimum-Mobile-CDS.

4.5.2. Performance of Stable-Mobile-CDS Look-ahead

When the look-ahead window size is 0, we obtain the performance of Minimum-Mobile-CDS. As we increase the look-ahead window size, algorithm Opt-CDSTrans becomes increasingly stability-conscious and yields the sequence of connected dominating sets that are more stable. For the simulation conditions tested, the value of $A_{critical}$ is 5 (refer Fig. 25, high-mobility scenarios) to 20 (refer Fig. 24, low-mobility scenarios). The decrease in the number of CDS transitions at $A_{critical}$ is by a factor 4–7, compared to that incurred at $A = 0$, while the relative increase in the average CDS size is only at most 20%.

As we increase $A$ from $A_{critical}$ to $A_{max}$, the relative decrease in the number of CDS transitions is 50–100%; while the relative increase in the CDS size (refer Fig. 26) is 15% (for 50 nodes network) to 100% (for 150 nodes network). The results indicate that it is possible to achieve a reasonably good level of stability (within 50 to 100% more than that of the optimal), without significant increase in the CDS size. Only when we aim for an absolute minimum for the CDS size or the CDS transitions, we incur a significant increase in the other performance metric.

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

In this section, we 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 packet. We are currently working on analyzing the performance of the stable path routing protocols and strategies with respect to other performance metrics like the node lifetime and jitter.

5.1. Energy consumption

In [17], we studied the impact of the tradeoff between stability and hop count in minimizing energy consumption. We wanted to explore the fact that for a given node mobility condition, as the

Table 4
Simulation parameters for Algorithm OptCDSTrans

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network size</td>
<td>$1000 \text{ m} \times 1000 \text{ m}$</td>
</tr>
<tr>
<td>Number of nodes</td>
<td>50, 100 and 150</td>
</tr>
<tr>
<td>Simulation time</td>
<td>1000 s</td>
</tr>
<tr>
<td>Transmission range</td>
<td>250 m</td>
</tr>
<tr>
<td>Topology sampling interval</td>
<td>1 s</td>
</tr>
<tr>
<td>Algorithms/heuristic</td>
<td>Heuristic $d$-MCDS and Algorithm OptCDSTrans (run using complete knowledge of future topology changes, prediction with uncertainty, look-ahead window size)</td>
</tr>
<tr>
<td>Minimum node speed</td>
<td>0 m/s</td>
</tr>
<tr>
<td>Maximum node speed</td>
<td>10, 15, 20, 30, 40 and 50 m/s</td>
</tr>
<tr>
<td>Pause time</td>
<td>0 s</td>
</tr>
</tbody>
</table>

Fig. 26. Number of Nodes per CDS for Stable-Mobile-CDS Look-ahead.
network density increases, aiming for a Stable-Mobile-Path reduces the number of route transitions at the cost of an increased hop count. On the other hand, as the network density increases, aiming for a Minimum-Hop-Mobile-Path reduces the hop count per path, but results in increased number of route transitions. 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. We now present a few significant results from [17]:

- As the energy consumed per hop for data packet transfer is reduced (i.e., as we adopt reduced overhearing or no overhearing models), a Stable-Mobile-Path can bring significant energy savings compared to that obtained when using a Minimum-Hop-Mobile-Path.

- When data packets are sent continuously but at a reduced rate, it is better to use a long-living stable path. If minimum hop paths are used, we may end up discovering a route to send every data packet, nullifying the energy savings obtained due to a reduced hop count.

- At high packet rates (i.e., high 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 gets counted as a one-time investment and is overshadowed by the energy spent in packet transfer, which is a regular expenditure.

- At low network density, energy consumption overhead due to flooding is relatively less due to the reduced number of retransmissions. At high network densities, route discovery is more expensive in terms of energy consumption. As a result, to minimize the overall energy consumption at moderate traffic, minimum-hop based routing is to be preferred at low network densities and stability-based routing is to be preferred at high network densities.

5.2. End-to-end delay per packet

In [15], we studied the performance of stable path routing protocols like ABR, FORP and RABR in \(ns-2\) and measured the stability of the routes chosen by these protocols and the end-to-end delay per packet for a source–destination session running each of these three protocols. We observed a stability-hop count tradeoff within the class of stability-based routing protocols: For the simulation conditions tested in [15], 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 \(v_{\text{max}} \leq 30 \text{ m/s}\), 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 in the case of FORP. Especially at high offered load (high data traffic), 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.

6. Conclusions and future work

The high level contribution of this paper is algorithms \(\text{OptPathTrans}, \text{OptTreeTrans}\) and \(\text{OptCDS-Trans}\) to generate the sequence of stable paths (Stable-Mobile-Path), the sequence of stable multicast Steiner trees (Stable-Mobile-Multicast-Steiner-Tree) and the sequence of stable connected dominating sets (Stable-Mobile-connected-dominating-set), respectively, over a time period of a network session. The sequence of minimum hop paths, minimum edge multicast Steiner trees and MCDSs that exist during the same time period are referred to as Minimum-Hop-Mobile-Path, Minimum-Mobile-Multicast-Steiner-Tree and Minimum-Mobile-connected-dominating-set, respectively. We study the performance of the three algorithms when the complete knowledge of future topology changes is available at the time of path/tree/CDS selection. We observe a distinct tradeoff between path hop count and the number of path transitions, number of edges in the Steiner tree and the number of Steiner tree transitions, number
of nodes in the CDS and the number of CDS transitions. It is highly impossible to simultaneously achieve optimality in the above mentioned contrasting performance metrics for paths, trees and CDS.

We also study the performance of the three algorithms under the following two models that predict locations of nodes in the future: (i) prediction with uncertainty and (ii) look-ahead window. Though we do not get the minimum number of transitions, the sequence of stable paths, trees and CDS generated by running the three algorithms under the Prediction with Uncertainty model are highly stable compared to their minimum mobile versions. At the same time, the hop count, number of edges and the number of nodes in these stable paths/trees/CDS respectively is not as high as that observed in the stable mobile paths/trees/CDS obtained when the algorithms are run with complete knowledge of the future topology changes. When the three algorithms are run under the Look-ahead window model, we observe that there is a critical look-ahead window size \( A_{\text{critical}} \), below which there is a maximum decrease in the number of path/tree/CDS transitions and a minimum increase in the hop count per path/edges per tree/nodes per CDS. As we increase \( A \) beyond \( A_{\text{critical}} \), we observe a relatively smaller decrease in the number of path/tree/CDS transitions, but a larger increase in the hop count per path/edges per tree/nodes per CDS.

Note that the Dijkstra algorithm, Kou et al. and \( d-MCDS \) heuristics are merely used as a tool to find the appropriate stable communication structures. The optimal number of route/tree/CDS reconstructions does not depend on these underlying algorithms as we try to find the longest living path/tree/CDS in the mobile sub graph spanning a sequence of static graphs. But, the run-time complexity of each of the proposed algorithms depends on the underlying algorithm used to determine the stable mobile path/Steiner tree/CDS. We are currently working on the distributed versions of \( \text{OptPathTrans}, \text{OptTreeTrans} \) and \( \text{OptCDSTrans} \) and are considering various location-update and mobility prediction mechanisms (e.g. \([12,20]\)) that can be used to gather and/or distribute knowledge of future topology changes.

References


Natarajan Meghanathan is an Assistant Professor in the Department of Computer Science at Jackson State University, since August 2005. He graduated with a Ph.D. in Computer Science from University of Texas at Dallas (UTD). His Ph.D. Dissertation was in the area of Mobile Ad hoc Networks (MANETs). His areas of expertise and interest include wireless and mobile computing, computer algorithms, computer networks, operating systems, distributed computing and network security. He has peer-reviewed publications in several leading journals and conferences. He won the best paper award in the 2006 ACM Southeast Conference at Melbourne, FL. He is a Sun Certified Java Programmer since February 2001 and was a Cisco Certified Network Associate from April 2001 to 2004. He completed a Graduate Information Assurance program at UTD in 2004. At JSU, He teaches Programming Fundamentals in C++/Java, undergraduate and graduate Computer Networks and Wireless Networks courses. Besides, he advises graduate students for their Masters Theses. He chaired the Networking and Distributed Systems track at the IEEE Southeast Conference in Memphis, 2006. He is serving as a Program Committee member of the following conferences: IEEE Wireless Communications and Networking Conference 2007, IEEE Advanced International Conference on Telecommunications 2007, IASTED Parallel and Distributed Computing 2007 and International Conference on Communication, Internet and Information Technology 2006. He has also reviewed papers for the following journals: Journal of Wireless Networks, The Computer Journal, Journal of Wireless Communications and Networking.

Andras Farago is currently working as a Professor of Computer Science at University of Texas at Dallas (UTD). He received the M.Sc. and Ph.D. in Electrical Engineering from the Technical University of Budapest, in 1976 and 1981, respectively. In 1996, he obtained the distinguished title Doctor of the Hungarian Academy of Sciences. He serves as editor for the Journal of Wireless Networks. His research focuses on algorithms, protocols, design, and analysis methods of communication networks. He has authored over a hundred papers in his research area.