Hybrid Position-Based Routing Algorithms for 3D Mobile Ad Hoc Networks

Song Liu, Thomas Fevens, and Alaa Eddien Abdallah
Department of Computer Science and Software Engineering
Concordia University, Montréal, QC, Canada, H3G 1M8
Email: pinestay@msn.com,{fevens,ae_abdal}@cse.concordia.ca

Abstract

Numerous routing algorithms have been proposed for routing efficiently in mobile ad hoc networks (MANETs) embedded in two dimensional (2D) spaces. But, in practice, such networks are frequently arranged in three dimensional (3D) spaces where the assumptions made in two dimensions, such as the ability to extract a planar subgraph, break down. Recently, a new category of 3D position-based routing algorithms based on projecting the 3D MANET to a projection plane has been proposed. In particular, the Adaptive Least-Squares Projective (ALSP) Face routing algorithm (Kao et al.*, 2007) achieves nearly guaranteed delivery but usually discovers excessively long routes to the destination. Referencing the idea of hybrid Greedy-Face-Greedy (GFG) routing in 2D MANETs, we propose a local hybrid algorithm combining Greedy routing with ALSP Face routing on projection planes. We show experimentally that this hybrid ALSP GFG routing algorithm on static 3D ad hoc networks can achieve nearly guaranteed delivery while discovering routes significantly closer in length to shortest paths. The mobility of nodes is handled by introducing the concepts of active sole nodes and a limited form of flooding called residual path finding to the ALSP GFG routing algorithm. To further analyze the performance of this mobility-aware hybrid algorithm, we define a mobility model for 3D MANETs. Using this mobility model, we demonstrate that the mobility-adapted hybrid routing algorithm can maintain high delivery rates with decreases in the average lengths of the paths discovered compared to shortest paths, without generating a large amount of flooding traffic.*

1. Introduction

A wireless ad hoc network is a collection of wireless hosts that can communicate without a fixed infrastructure. Each host in the network can communicate with all other hosts within its transmission range [4, 5], which we will assume for this paper to be a fixed range R for all hosts. If two hosts are not able to communicate directly then a multi-hop routing protocol is needed for the hosts to send packets to each other. Since the wireless hosts that we are modeling commonly have only have a limited power supply (such as a battery) and a limited amount of memory, particularly for ad hoc sensor networks, local algorithms are typically preferred. A local algorithm uses only the information on neighboring nodes up to a fixed number of hops away. We are specifically interested in *position-based* routing protocols where a node forwards packets using the location (coordinates in the plane or space) of itself, its neighbors (up to a fixed number of hops away), and the destination [13].

Many routing algorithms have been proposed and studied extensively in the context of two dimensional (2D) spaces during the past few years [13, 22]. But in the real world, e.g., in modern high-rise buildings, in rough terrain, or in outer space, hosts are naturally positioned in a three dimensional (3D) space where some assumptions made in 2D, such as the ability to extract planar subgraphs, break down. For example, in underwater acoustic sensor networks, sensors may be deployed to monitor and report on phenomena within an underwater volume [24]. Thus for practical applications, routing algorithms for 3D ad hoc networks should also be considered. Ideally, such a local routing algorithm should guarantee delivery without resorting to flooding, but as shown by Durocher *et al.* [10], it is not possible to have a local routing algorithm with guaranteed delivery if the nodes of the 3D ad hoc network are contained in a slab of thickness greater than $1/\sqrt{2}$. Therefore, one objective of this paper is to present local position-based routing algorithms on unrestricted static 3D ad hoc networks that have nearly guaranteed delivery with discovered routes being close to the shortest paths on average.

As an added complication, the hosts in a 3D ad hoc network may be mobile, with wireless mobile hosts possibly moving to new positions during routing. In a mobile ad hoc network (MANET) the underlying topology of the whole network is usually not available and may be variable. However, most of routing algorithms in MANET that were developed in recent years assume that all the mobile hosts are stable (or that the motion is negligibly small compared to routing time) during the routing. Under this condition, forwarding a packet in MANET is equivalent to routing on a static topology, which may not represent some real-world scenarios. Thus a second objective of this paper is to create a mobility model of a 3D MANET and determine the influences of the mobility environment on position-based routing algorithms. Based on these influences, modification are proposed to improve the performance of the new routing protocols on a MANET.

2. Preliminaries

Let the set of n wireless hosts in our MANET be represented by a set of nodes S in the 3D Euclidean space $\mathbf{R}^3$, each node possessing a geometric location. Since all the network hosts have the same communication range R, this maximum transmission range can be represented as a sphere volume of radius R with the node at its center. Two nodes are connected by an (undirected) edge if the Euclidean distance between them is at most R. The resulting (Euclidean) graph is called a (3D) unit ball graph (*UBG*).

2.1. Routing Algorithms

For *position-based* routing (also known as online routing [7, 6] or geographic routing [15]), the position of the nodes can be obtained, for example, using GPS. There are many ways to use position information to assist in making routing decisions. In one class of algorithms, *progress-based* algorithms as categorized in [23], the algorithm forwards the packet in every step to exactly one of its neighbors, which is chosen according to a specified heuristic which aims to choose a neighbor that makes progress (however it is defined) to the destination.

To analyze the performance of our routing algorithms, we will use two metrics, the delivery rate and the hop stretch factor. The *delivery rate* is defined as the percentage of successful deliveries to the destination. The *(hop) stretch factor* is defined as ratio of the number of hops traversed by using the routing algorithm over the least number of hops between the source and the destination.

Progress-based Routing. In GREEDY routing [11, 27], or 3D GREEDY routing [16], a node forwards the packet to its neighbor which is closest to the destination. COMPASS or *directional* routing [21], or 3D COMPASS [16], moves the packet to a neighboring node such that the angle formed between the current node, next node, and destination is minimized. Other 3D progress routing algorithms are 3D ELLIPSOID and 3D MOST FORWARD routing algorithms [16] based on the 2D algorithms of the same names [29, 27]. As with the 2D versions of these algorithms, all of these types of progress-based algorithms fail to deliver the packet in certain situations.

Two Dimensional Routing. In two dimensions, to overcome the inability of progress-based routing to always deliver their packets, position information can be used to extract a *planar* subgraph (which contains no edges that cross) such that routing can be performed on the faces of the subgraph, known as *face* routing or perimeter routing [21, 7]. The advantage of this approach is that delivery of packets can always be guaranteed.

One drawback of face routing is that, although delivery is guaranteed on a planar graph, the route discovered may be many times longer than a shortest path in the graph. To try to reduce the discovered path length, [9] (also see [26]) and [18] use a hybrid routing algorithm that combines greedy routing with face routing. The idea is to use greedy routing until a local minimum (where all the one hop neighbors of the current node c are further way from the destination node than c) is reached whereupon face routing is used. After the local minimum is bypassed, the algorithm switches back to greedy routing, and so on. This algorithm is termed GFG (Greedy-Face-Greedy) by [9] and GPSR (Greedy Perimeter Stateless Routing) by [18].

Three Dimensional Routing. Face routing and GFG hybrid algorithms guarantee delivery only over a 2D planar geometric graph. In a 3D network, we cannot directly perform the face routing protocol since since the notions of planarity and routing about the perimeters of faces do not exist. To enable routing based on the face routing protocol, Kao *et al.* [16] propose mapping the 3D network to a projection plane where the 2D face routing algorithm can then be applied. Specifically, their PROJECTIVE FACE algorithm uses two orthogonal planes that intersect along the line between the source and the destination, changing to the second projection plane if the first plane leads to failure. This algorithm gives significantly better delivery rate than the other deterministic 3D routing algorithms such as 3D GREEDY, 3D COMPASS, 3D ELLIPSOID and 3D MOST FORWARD routing algorithms although the algorithm also leads to a very large hop stretch factor: e.g., about 93% and 9.5, respectively, on a sample simulation of 75 nodes randomly generated $10,000$ times with $R = 25$ in a $100 \times 100 \times 100$ cube, running the algorithms once per network to obtain the average delivery rates and stretch factors.

To enhance the performance of the PROJECTIVE FACE routing algorithm, in [17], Kao *et al.* propose three heuristics to modify the PROJECTIVE FACE routing algorithm. The main idea of these heuristics is the following. Using the current node, its neighbors up to two hops away, and the destination node, an initial projection plane (the LSP plane) by using least-squares error minimization of the distances of the nodes to the plane. The network is projected to this LSP plane for PROJECTIVE FACE routing. To ensure that this LSP plane remains appropriate for the current node, after a fixed number of hops (using an ABS threshold value) during routing, the LSP plane is recomputed. The last heuristic involves using N_s projection planes arranged in a fixed order about an axis. The algorithm switches between these planes, following the order, to disrupt any looping that may occur during routing. The resulting 3D routing algorithm, called ADAPTIVE LEAST-SQUARES PROJECTIVE (ALSP) FACE routing, gives nearly certain delivery rate, e.g., nearly always 100% by simulation, while increasing the hop stretch factor (to at worst about 15 in our sample simulation).

Several other 3D position based routing algorithms have also been proposed. In [1] Abdallah *et al.* represent a simple heuristic that uses a projection approach to adapt face routing to the 3D environment. The algorithm called CFACE(3) (Coordinate Face), in which all the nodes are projected in the $x-y$ plane, then the algorithm performs the Face routing on this projected graph. If the packet cannot reach the destination, CFACE(3) then projects all the nodes in the $y-z$ plane; if it fails again, the $z-x$ plane is used. CFACE(3) fails if the face routing could not reach the destination on the third projection plane. In our sample simulation, the delivery rate of this algorithm was about 95%, but with hop stretch factor above 10. If the nodes of the 3D ad hoc network are contained in a slab of thickness not greater than $1/\sqrt{2}$, Durocher *et al.* [10] present a local routing algorithm based on face routing on a projected plane with guaranteed delivery.

Position information can also be used to the amount of traffic used by flooding-based routing algorithms. For instance, in Distance Routing Effect Algorithm for Mobility (DREAM) [5] and the geocasting-based Location-Aided Routing (LAR) [19, 2], information about the position of the destination is used to limit the extent of flooding. Nodes whose position makes it unlikely for them to be on a shortest path to the destination (outside the flooding region) will simply not forward packets. In 3D, both DREAM and LAR use a spherical *expected region* about the destination node d of radius $v_{max} * (t_1 - t_0)$ where t_1 is the current time, t_0 is the time stamp of the position information that c has about d and v_{max} is the maximum speed of d. DREAM defines a flooding region of a cone with its apex at the source node, centered on the destination node, with the minimum angle to contain the entire expected region in its interior. LAR defines a flooding region of the minimum-size cube with the source node in one corner and the expected zone in the opposite corner. DREAM and LAR reduce the flooding traffic compared to general flooding, but it still very high compared to progress-based routing.

2.2. 3D Geometric Subgraphs

In this paper, we also consider the behavior of the routing algorithms on several subgraphs of the unit ball graph. Several geometric subgraphs have been proposed using local information in two dimensions. The Gabriel graph (*GG*) and relative neighborhood graph (*RNG*) are two well-known subgraphs that are extracted from the *UBG*. Kao et al. in [16] proposed schemes to use *GG* and *RNG* algorithms to extract equivalent subgraphs in 3D. In the following, define $S(c, r)$ to be a sphere centered on c of radius r. Define $d(x, y)$ to be the Euclidean distance between x and y. Let $G =$*UBG* be a 3D unit ball graph.

3D Gabriel Graph. Consider the sphere centered at the midpoint between the points u and v with radius $d(u, v)/2$, $S((u+v)/2, d(u, v)/2)$. Then the *3D Gabriel Graph* of G, denoted *3D GG*(G), is defined as follows, based on the definition of the 2D *Gabriel Graph* [12]. Given any two adjacent nodes u and v in G, the edge (u, v) belongs to *GG*(G) if and only if no other node $w \in G$ is located in $S((u+v)/2, d(u, v)/2)$.

3D Relative Neighborhood Graph. The *3D Relative Neighborhood Graph* of G, denoted *3D RNG*(G), is defined as follows, based on the definition of the 2D *Relative Neighborhood Graph* [28]. Given any two adjacent nodes u and v in G, the edge (u, v) belongs to *3D RNG*(G) if and only if no other node $w \in G$ is located in the lens region defined by the intersection of $S(u, d(u, v))$ and $S(v, d(u, v))$.

2.3. Mobility Models for MANETs

The movement pattern of mobile hosts plays an important role in MANET. The most efficient method to evaluate a routing protocol is through a simulation. The topology and movement of mobile nodes are the key factors in the performance of routing protocols. Once the nodes are initially placed, the mobility model dictates where each mobile node moves to. A variety of mobility models have been proposed for MANET. All of these mobility models have nodes that randomly choose a direction between 0 and as well as a speed from a given range. In [3], in order to emulate a "natural" movement of the nodes in 2D or 3D, beside randomly generating speed and movement angle, the authors provide one of two original parameters (ascertain amount of time elapsed on distance is traveled, respectively): time-to-live and distance-to-live. After expiration of either one of the two parameters, each node receives new movement parameters (speed, direction, and time-to-live or distance-to-live). In the Random Walk mobility model [14], when nodes move to a specified step or a time period, the nodes repeat this process. The Random Direction model [25] operates similarly to the Random Walk model, except that nodes keep moving until they reach the simulation boundary, where these nodes then choose a new direction and a new speed. The random Waypoint model [8] also has a similar procedure, except that when nodes reach their selected destination, they suspend motion for a while (what is known as "pause time") and then select a new direction and a new speed.

With mobility, several routing issues arise such as mobile destinations. Just as any node may move during routing in a MANET, the destination may have moved from its initial position during routing. Kranakis *et al.*[20] deal with the the problem of mobile destinations by flooding to the nodes within a circular region around the destination's initial position, the radius of the region determined by the destination's maximum velocity. Alternatively, Kranakis *et al.* propose traversing a tree of all the faces intersecting or contained within the circular region.

3. New 3D Routing Algorithms

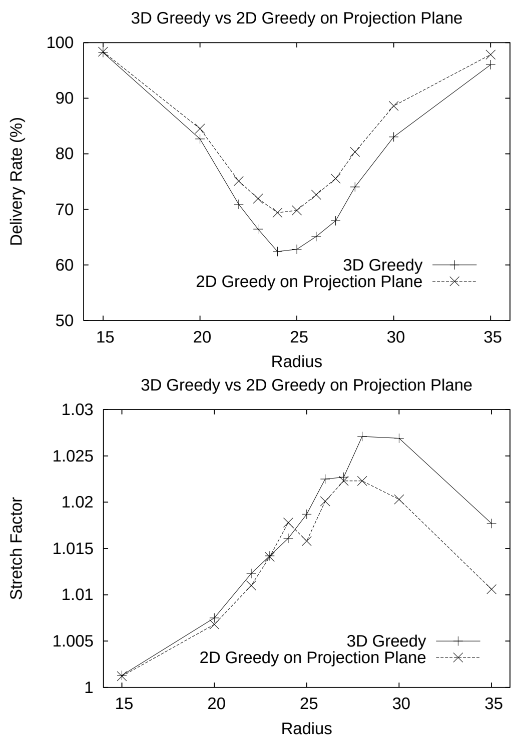

Figure 1. Comparison of 3D GREEDY **on the** *3D UBG* **versus** 2D GREEDY **on a projection of the graph onto a LSP plane. Top: Delivery Rates; Bottom: Stretch Factors.**

Although the delivery rates of 3D GREEDY are relatively low, when successful, it finds nearly the shortest path (i.e., with a stretch factor near 1.0). On the other hand, the Adaptive Least-Square Projection Face (ALSP FACE) routing algorithm can almost provide a nearly guaranteed delivery rate but with a large hop stretch factor. We have two general approaches to hybridizing 3D GREEDY with the ALSP FACE routing algorithm to try to benefit from both algorithms. One option would be to use 3D GREEDY until we arrive at a local minimum, then project the network to an ALSP plane and continue with ALSP FACE routing until the current node enters a switching zone, whereupon we return to the network in the 3D space and continue with 3D GREEDY routing. The second option is that we first project the network onto a LSP plane and then perform 2D GREEDY routing and ALSP FACE routing on this 2D projection plane. To determine which option to use, we first compare the performance of 3D GREEDY routing to that of 2D GREEDY routing (the latter on a LSP plane). From the results presented in Fig. 1, we find that 2D GREEDY routing on a projection plane achieves a better performance both in terms of delivery rate and hop stretch factor. Therefore, for our hybrid routing algorithms we will preform both GREEDY routing and ALSP FACE routing on the ALSP planes.

3.1. ALSP GFG Routing Algorithm

The Adaptive Least-Squares Projective Greedy Face Greedy (ALSP GFG) routing algorithm starts with the GREEDY algorithm on the first order ALSP plane. Upon reaching a local minimum l, it switches to the ALSP FACE routing algorithm until the current node is closer to destination d than l and the node becomes a switch node (a node where the hybrid routing algorithm switches from FACE routing to GREEDY routing). In essence, the current node has entered a *switching zone* that is defined by a circle centered at d with radius $d(l, d)$ as in the original 2D GFG routing algorithm. When the current node enters the switching zone and becomes a switch node, GREEDY routing resumes. ALSP GFG also incorporates all the loop detection mechanisms described below (visited edges array, restart nodes, and switch node array) as well as the heuristics of the ALSP FACE algorithm (least-squares ALSP plane, multiple orders of ALSP planes, and ABS).

Motivated by exclusion regions used in creating subgraphs such as the Gabriel Graph, we have also experimented with other geometric shapes for the switching zone such as cones, ellipses, and smaller circles adjacent to the current node. While these shapes for the switching zone resulted in lower stretch factors for the hybrid routing algorithms, the trade-off was lower delivery rates. Therefore, we did not include these variations of ALSP GFG in this paper.

3.2. Loop Detection

One of the issues in developing hybrid GFG routing algorithms is to address the problem of looping which leads to large stretch factors if detected late and potentially lead to failure. In the implementation of ALSP GFG, we propose several mechanisms to deal with looping to complement the heuristics developed for the PROJECTIVE FACE routing algorithm [17] (i.e., the least squares projection plane, the N_s related projection planes and the adaptive behavior scale (ABS)). In developing these additional mechanisms, we seek to detect and break out of loops as soon as possible while avoiding quickly re-entering into a related loop. The main strategy for breaking out of a detected loop is to switch to the next order ALSP projection plane.

Visited Edges Array. We define an array of size M that saves the last M visited edges as the hybrid algorithm performs. We'll call this the *visited edges array*. If the current visited edge is present in the visited edges array, the routing algorithm assumes it has entered into a loop. The routing algorithm then switches to the next order projection plane.

Restart Nodes. The visited edges array also serves a second function. We use this array to determine which of the past M visited nodes has the highest degree to determine a *restart node*, defined below. If the hybrid routing detects a loop through the visited edges array, before it calculates the next order projection plane, the algorithm backtracks to the node in the visited edges array with the highest degree (these additional hops will be considered to be part of the routing). The starting point on the next order LSP plane will be this higher degree node, the restart node. The motivation for this strategy is that since the routing algorithm will start performing GREEDY routing on the new ALSP plane starting at this restart node, with a higher degree there will be less chance of immediately encountering a local minimum.

Switch Node Array. The second loop is related to the hybrid routing algorithm itself. When GREEDY routing arrives at a local minimum node s, the hybrid routing algorithm switches to face routing. We'll call s a switch node. In some situations, the hybrid algorithm may encounter a sequence of switch node, eventually arriving back at a previously encountered switch node. If a switch node is visited twice, the hybrid routing algorithm deems that the current routing is performing inside a loop. To detect this type of loop, we use an array of size K, called a *switch node array*, to memorize the K most recent switch nodes encountered. If the current switch node is present in the switch node array, the hybrid routing algorithm switches to next order LSP plane starting from the restart node. Note that loops longer than the size of visited edges array may still be detected by the switch node array.

All of these mechanisms aim to reduce the stretch factor. At the same time, they may also decrease the delivery rate by increasing the chance of using up all N_s LSP planes and then failing.

4. Routing on 3D Mobile Geometric Graphs

In this section, we will present our mobility model and modifications to the routing algorithms to improve their performance under mobility.

4.1. 3D Random Curve Mobility Model

Referring to the previous mobility models [3, 8, 14, 25] discussed in Section 2.3, and adapting them to the 3D network environment, we propose a new 3D mobility model which we call the *3D Random Curve Mobility Model*.

Firstly, we determine a mobility domain where the nodes of the network are confined. The mobility domain will be an axis-oriented cube in three dimensions with one corner at the origin and the corner opposite at (L, L, L). To create a MANET, we randomly generate N nodes in this cube. Each node has an initial position (a_0, b_0, c_0) at a time $t = 0$, where a_0, b_0, and c_0 are in the range from 0 to L.

Secondly, to define a mobility curve to calculate the new position $(x(t), y(t), z(t))$ of each mobile node at times $t > 0$, we use the following three polynomials to determine its trajectory:

$$x(t) = a_0 + a_1(t-l) + a_2(t-l)^2 + a_3(t-l)^3$$
$$y(t) = b_0 + b_1(t-m) + b_2(t-m)^2 + b_3(t-m)^3$$
$$z(t) = c_0 + c_1(t-n) + c_2(t-n)^2 + c_3(t-n)^3$$

In these polynomials, the coefficients a_i, b_i, and c_i ($1 \leq i \leq 3$) are assigned values randomly from -1 to 1. The coefficients l, m, and n, which are initially 0, are used to continue the trajectory of a node when it hits the boundary.

To control the velocity of the node we use a fixed time step Δt to determine the next t value. After a fixed number of some time steps (for our simulations we will use 10 time steps), the simulation will randomly renew all the a_i, b_i, and c_i ($1 \leq i \leq 3$) coefficients in the above polynomials, whereas a_0, b_0, and c_0 are updated to the current position of the node. If a new position of a node at any time step results in the node exiting the cube, the coefficients of the polynomial are adjusted to make the node bounces back off the boundary of the cube to make sure the node remains inside the cube. If the coefficients for a node are being updated at the same time step as the node is bouncing off the boundary of the cube, updating the coefficients of the trajectory for this node is delayed.

4.2. Issues Arising from Mobility

In this section, we address issues that arise in a MANET such as what to do when an actively routing nodes temporarily has no neighbors, and how to deal with destinations that have moved away from their initial routing positions at the start of routing. For our mobility model, we assume that the time for a routing algorithm to make one hop is equal to the time step Δt of the above mobility model. In addition, we ignore the time cost of broadcasting and receiving a response from its neighbors during routing, and will assume that when the current node intends to transmit a message, it has the latest neighbor list.

Active Isolated Nodes. In the mobility environment, each node randomly moves to a new position at each time step. So, upon receiving a packet, a node moves to a new position. In some cases, because of its movement, this node may become an isolated node with no neighbors. If we just simply consider that the routing algorithm could not reach the destination when it encounters an isolated node (we'll term this the *isolated node failure* strategy), the delivery rate then does not reflect the actual possible performance of this routing algorithm.

To improve the actual performance of routing protocols in the mobility environment, we need to accommodate isolated nodes during the routing. In our solution when the node that holds a packet becomes an isolated node, we make this node hold the packet until it has at least one neighbor. We call such a node an *active isolated node*. For example, suppose an active isolated node has to wait five time steps until it again has a neighbor. We deem that the path for the current packet has extra five hops during routing (one hop per time step).

Residual Flooding Protocol. To deal with the issue of mobile destinations, rather than declaring a routing failure if the destination could not be found at its initial position within the transmission range of the current node, we using a constrained flooding-based approach to try to guide the packet to the current location of the destination. In our approach we don't assume we know anything about the destination other than its initial position at the start of routing and its identification number (we don't know its velocity, for example, and thus cannot predict how far the destination may have traveled). Initially ALSP GFG forwards the packet to a node that would be expected to have the destination as a neighbor. If the destination cannot be found in this region because it has moved, the routing algorithm switches to a constrained version of flooding, which we term *residual path flooding*, to broadcast the packet. Only those nodes receiving the flooded packets that either have, or have had, the destination node as a neighbor will broadcast the packet to its transmission region. If a node that has not seen the destination receives this packet, it will discard this packet. Clearly, the flooding then chooses a direction (the direction that the destination has moved) to broadcast the packet. When the flooding is finished and the destination still could not be found, this routing mission is deemed to have failed.

4.3. Performance Measures

In this section, we further consider the implications of mobility on our performance measures. Specifically, in terms of measuring the stretch factor for routing, what modification do we need to make to the definition of a shortest path. Also, we introduce a flooding rate measure to account for the additional impact of the residual path flooding protocol.

The Shortest Paths. In our static environment, we mainly use stretch factor and delivery rate to reflect a routing protocol's performance. We also want to adopt these two metrics in the mobility environment. We could use the same method as we did for static simulation to calculate the delivery rate. However, determining the stretch factor in mobility environment becomes a challenge because each node does not have a stable position.

In a static network, we use the ratio of the number of hops during routing to that for a shortest path from the source to the destination to determine the stretch factor. We note that we can perform flooding to determine the length of a shortest path in a static environment. In mobility, at each time interval, each mobile node moves to a new position, and we cannot predict how the topology will change during the movement of the nodes. A shortest path is also variable, changing at different times. Because of this reason, the hop number sometimes may be shorter (or longer) than that of an initial static shortest path (if the nodes remained static). In some scenarios, because of the node movement, the largest connected component may divide into several parts with the source and destination located in different disjoint components. But, after a few time steps of movement, the destination may be reachable again.

Based on the above characteristics, we propose the following scheme to determine a shortest path in a mobility environment. Firstly, we determine the largest connected component as in the static case (at the time $t = 0$), and randomly choose a source node and a destination node in this component. Secondly, with each time step, we move each node to its new position. Meanwhile, we perform the flooding to forward a packet from source to destination, one hop per time step. Whether a shortest path exists to the destination depends on the destination being *reachable*. We consider two cases where the destination is reachable. In the first case, assuming each node will receive and broadcast the packet to its current neighbors, if any, only once, the destination is a neighbor of the nodes that are actively spreading the packet during flooding. Then, a shortest path would be defined as the fewest number of the hops that a packet makes to reach the destination. In the second case, the destination is still reachable even though flooding could not deliver the packet to it. Consider a node q which receives the packet after x hops during flooding. Now, suppose that after y more time steps (and the flooding has without encountering the destination) that q becomes a neighbor of the destination (whether because the destination has left the connected component, or not). In this case, we define the shortest path to the destination to be x+y hops. Even though flooding perhaps might not find the destination later, we still deem, at this moment, the current graph as connected between the source and destination. Obviously, if a routing algorithm arrives at q after x+y-1 hops along some path, then it would reach the destination on the next hop.

Flooding Rate. In the mobility environment, we combine single path and residual path flooding to delivery a packet. Clearly, flooding potentially requires many more hops to forward a packet than a single path strategy. Therefore, we need to ascertain the percentage of hops used during the flooding phase to the whole hop count during routing to gauge the impact of the flooding component. To determine this percentage, during the run times, we only record the flooding hop counts and total hop numbers of the routings that have successful delivered a packet. Then we use the ratio of the flooding hop counts to the total number of hops to determine a *flooding rate* (which we will represent as a percentage of the total number of hops).

5. Performance evaluation of routing algorithms

5.1. Simulation Environment

The following simulations were performed as follows. We randomly generate $N = 75$ nodes at positions (a_i, b_i, c_i) in a $100 \times 100 \times 100$ (i.e., $L = 100$) cube in 3D. Based on the static positions of each node, we calculate the largest connected component and randomly choose a source and a destination from it. We perform our routing algorithms on the graphs *UBG(S)*, *GG(UBG(S))* and *RNG(UBG(S))* based on the randomly generated sets of nodes S. We repeat the above a total of $100 * 100$ times to achieve the average experimental results.

In order to compare the performance of the GREEDY routing, ALSP FACE routing, and our hybrid routing algorithms in 3D, our hybrid routing algorithms also perform on the same ALSP planes with $N_s = 32$. We use ABS=125 as experimentally determined for the ALSP FACE routing algorithm in [17]. We use 10 for the size of the visited edges array since a larger size only slightly decreases stretch factors and to minimize the amount of time checking the array for loops (and to reflect the space constraints on typical MANET nodes and the packets being sent). Also, we use a switch node array of size 10. The *3D GG(UBG)* and *3D RNG(UBG)* subgraphs are first computed on the *3D UBG* and then projected on the ALSP planes. Note that any geometric subgraph such as *GG(UBG)* and *RNG(UBG)* extracted from the projected 3D UBG by using only local information may not be planar [16].

For our mobility simulations, we use the same experimental setup described above with a few modifications. Here, the transmission radius of each host is fixed at 25 units (when performance is typically the poorest in static networks, see Fig. 2). Based on the starting positions of each node, again we calculate the largest connected component and randomly choose a source and a destination from it. But if the destination is determined to be unreachable during the mobile shortest path calculation, the graph is discarded and a new graph is generated.

In the mobility environment, after the routing algorithm makes one hop, we update each host's position. To determine the influence of the node velocity on the routing algorithms, we vary Δt from 0 to 4. When the time step of a node is more than 4 then in one hop the node may travel entirely across the cube.

5.2. Performance of Hybrid Routing Algorithms on Static Networks

From Fig. 2, we find that the ALSP FACE and ALSP GFG routing algorithms almost guarantee the delivery in the UBG, GG(UBG), and RNG(UBG) for all transmission radii (recall from Fig. 1, the 2D GREEDY on the same projection planes can have a delivery rate as low as about 70%). Indeed, ALSP GFG has a slightly higher delivery rate than ALSP FACE. When the transmission radius is small, all the routing algorithms have a higher delivery rate due to the largest connected component only having a few nodes and thus the source and destinations are separated by only a few hops. At the other extreme, when the transmission radius approaches 35 units, the *UBG* becomes very dense with a very high average degree and again the source and destination are again only separated by a few hops. In terms of the hop stretch factor, ALSP FACE has a much higher stretch factor whereas ALSP GFG is closer to that of 2D GREEDY (Fig. 1).

5.3. Performance of Under Mobility

The stretch factor reflects the ability of the routing algorithm to find a short routing path. To understand the relative meanings of the stretch factor of a routing algorithm between static and dynamic environments, we compare average shortest paths in the graphs across different time step values between these two environments in Fig. 3. We can find that on average the shortest path in a dynamic environment is lower than that for the static environment. Some of this lower length comes from the destinations coming in contact with nodes along the static shortest paths and thus shortening the paths. As well, for larger time step values,

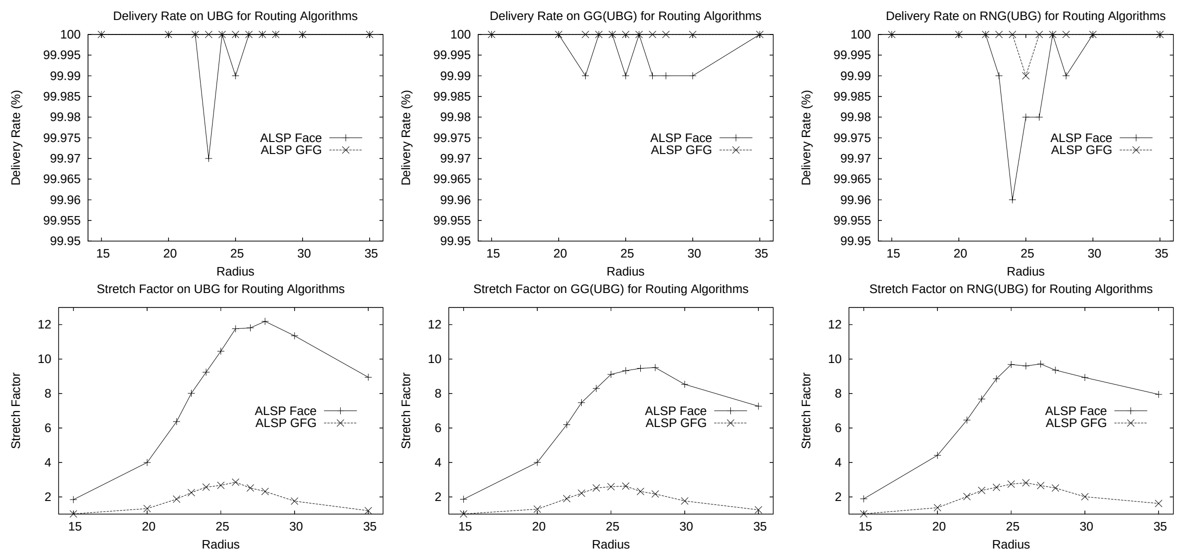

Figure 2. Delivery rates, top, and stretch factors, bottom, of ALSP GFG **and** ALSP FACE **algorithms on** *UBG* **(left column),** *GG(UBG)* **(center column) and** *RNG(UBG)* **(right column). Graphs are static.**

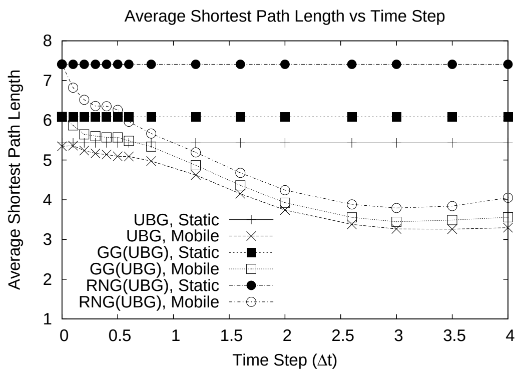

Figure 3. The average length of shortest paths of 1000 UBGs, and their 3D subgraphs, under static and dynamic conditions.

some of this lower average shortest path length is due to nodes bouncing off the boundary of the box domain.

In Fig. 4 we compare difference in performance of ALSP GFG using residual flooding and/or active isolated nodes. Using residual flooding has the greatest impact of increasing the delivery rate, especially as the time step gets large, while increasing the stretch factor. This is not surprising since without residual flooding, if the destination could not be found at its initial position within the transmission range of the current node, delivery failure would be declared. Using active isolated nodes has a lesser impact of also increasing the delivery rate while increasing the stretch factor. The lower impact is due to the relatively small number of isolated node encountered – typically less than 600 over 10,000 runs. Obviously, the best performance under mobility occurs when both residual flooding and active isolated nodes are used, which we will assume for the rest of the simulation results.

In Fig. 5, we consider the relative performance of selected routing algorithms using both residual flooding and active isolated nodes are used. To broaden the comparison sample, we also use a simple hybrid algorithm based on CFACE(3) [1], called G:CFACE(1):G which uses 3D GREEDY until it reaches a local minimum whereupon it uses CFACE(3), but only for one axis-oriented projection face, picked randomly, (hence called CFACE(1)) until it can resume 3D GREEDY. Among the algorithms considered, ALSP GFG consistently has the highest delivery rate across all time steps while its stretch factor is as good as any except for GREEDY. For low time step values, G:CFACE(1):G has the second best delivery rate, while for larger time step values, ALSP FACE has the second best delivery rate. For stretch factor, at low time step values, 3D GREEDY has the lowest values followed by G:CFACE(1):G and ALSP GFG. For larger time step values, all three algorithms have essentially the same stretch factors.

The flooding rate reflects the proportion of residual path flooding in the total number of transmission hops, for successful packet deliveries. From Fig. 6, the flooding rate is small when the time step is less than 0.75. This indicates that the algorithms do not perform residual flooding too often because the destination is still remaining inside its initial

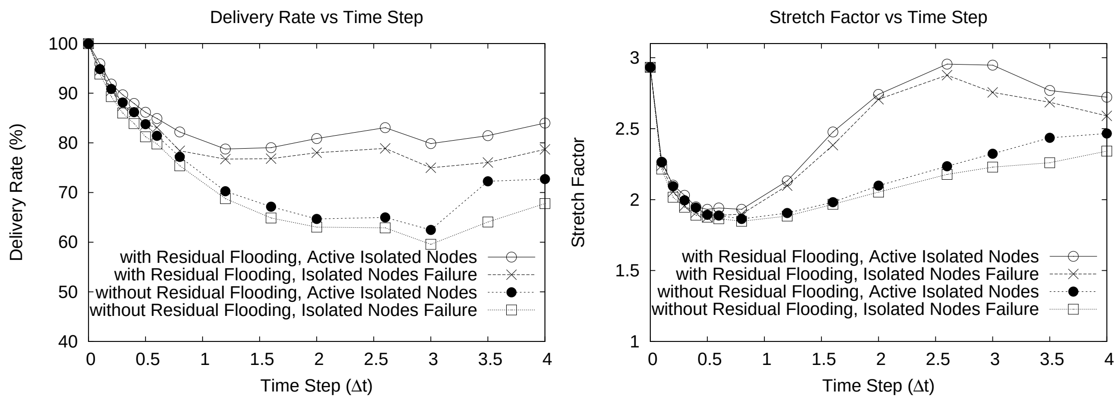

Figure 4. Comparison of the performance (Left: Delivery Rate; Right: Stretch Factor) of ALSP GFG **on** *UBG* **with, or without, Residual Flooding, and with Active Isolated Nodes or Isolated Node Failure.**

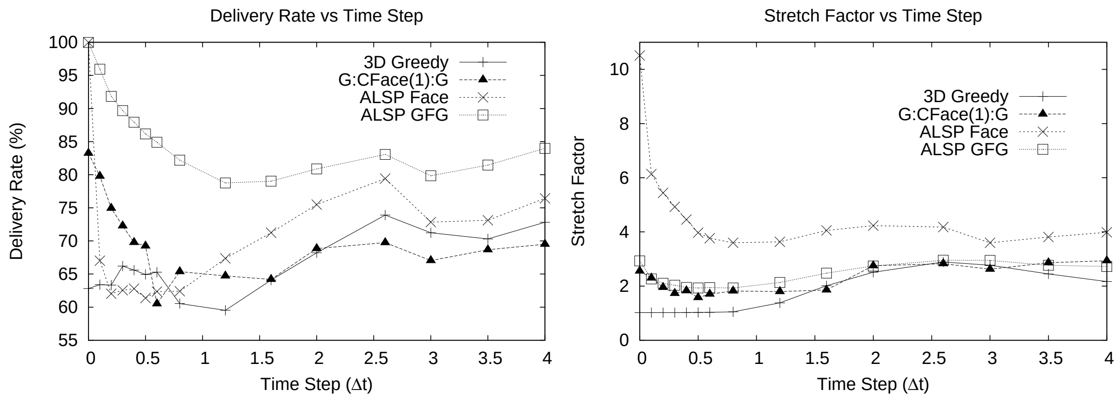

Figure 5. Comparison of the performance (Left: Delivery Rate; Right: Stretch Factor) of GREEDY**,** ALSP FACE**,** ALSP GFG**, and** G:CFACE(1):G **versus time step on** *UBG* **with Residual Flooding and Active Isolated Nodes.**

transmission range when a packet is forwarded to this area. Then, with the increase in time step (and correspondingly, moving speed), flooding become a large of proportion of the total transmission hops because the algorithms have to increasingly resort to using residual flooding to find the destination. When resorting to residual flooding, ALSP GFG and G:CFACE(1):G have the smallest percentage of packets due to flooding.

The delivery rates shown in Figs. 4 and 5 have a flexuous shape, decreasing then slightly increasing as the node velocity increases. This is because of the motion restriction we define for these moving nodes. We restrict all nodes to move inside a 3D cube in order to maintain the mobile node density so that we can compare the delivery rate with the experimental results from static stimulation. Following this rule, when the destination reaches the boundary of the cube, our random curve mobility model will make the destination reflect off this boundary. In some cases, this reflection could shorten the flooding procedure to find the destination. This can be seen in Fig. 6 where the percentage of packets during routing due to flooding increases until about $\Delta t = 2$ whereupon the percentage of packets due to flooding then decreases as the time step increases further.

6. Conclusions

In this paper, we present the hybrid routing algorithm ALSP GFG, based on projecting the 3D network to a set of planes, that achieve a performance in terms of delivery rate and stretch factor that is nearly on par with that of 2D hybrid routing algorithms short of guaranteeing delivery of the packet. As well, by using active isolated nodes and residual flooding, we can maintain high delivery rates under condi-

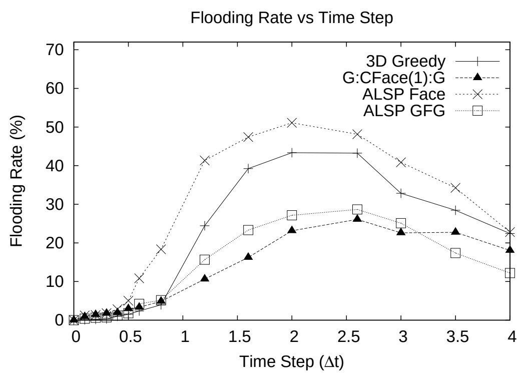

Figure 6. Comparison of flooding rates using Residual Flooding and Active Isolated Nodes for 3D GREEDY**,** ALSP FACE**,** ALSP GFG**, and** G:CFACE(1):G **versus time step on the** *UBG***.**

tions of node movement in dynamic ad hoc networks while avoiding the expense of using full flooding.

References

[1] A. Abdallah, T. Fevens, and J. Opatrny. Randomized 3-D Position-based Routing Algorithm for Ad-hoc Networks. In *Proc. of the 3rd MOBIQUITOUS*, San Jose, July 2006.

[2] A. Abdallah, T. Fevens, and J. Opatrny. High delivery rate position-based routing algorithms for 3d ad hoc networks. *Computer Communications*, 31(4):807–817, 2008.

[3] A. Abdallah, M. Hassan, G. Kao, and C. Morosan. Topology Control for Balanced Energy Consumption in Emergency Wireless Deployments. In *Proc. of PE-WASUN*, pages 41–48, Montréal, Canada, 2005.

[4] L. Barrière, P. Fraigniaud, L. Narayanan, and J. Opatrny. Robust position-based routing in wireless ad hoc networks with irregular transmission ranges. *Wireless Communications and Mobile Computing Journal*, 3(2):141–153, 2003.

[5] S. Basagni, I. Chlamtac, V. Syrotiuk, and B. Woodward. A Distance Routing Effect Algorithm for Mobility (DREAM). In *Proc. of the 4th MOBICOM*, pages 76–84, Dallas, 1998.

[6] P. Bose and P. Morin. Online Routing in Triangulations. In *Proc. of 10th ISAAC*, pages 113–122, 1999.

[7] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with Guaranteed Delivery in Ad Hoc Wireless Networks. *Wireless Networks*, 7(6):609–616, November 2001.

[8] J. Broch, D. Maltz, D. Johnson, Y.-C. Hu, and J. Jetcheva. Performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proc. of 4th MOBICOM*, pages 85–97, October 1998.

[9] S. Datta, I. Stojmenovic, and J. Wu. Internal node and shortcut based routing with guaranteed delivery in wireless networks. *Cluster Computing*, 5:169–178, 2002.

[10] S. Durocher, D. Kirkpatrick, and L. Narayanan. On routing with guaranteed delivery in three-dimensional ad hoc wireless networks. In *Proc. of ICDCN '08*, pages 546–557, 2008.

[11] G. Finn. Routing and addressing problems in large metropolitan-scale internetworks. Technical Report ISU/RR-87-180, USC ISI, Marina del Ray, CA, 1987.

[12] K. Gabriel and R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18:259–278, 1969.

[13] S. Giordano, I. Stojmenovic, and L. Blazevic. Postion Based Routing Algorithms for Ad Hoc Networks: A Taxonomy. In X. Cheng, X. Huang, and D. Du, editors, *Ad Hoc Wireless Networking*, pages 103–136. Kluwer, 2003.

[14] R. A. Guerin. Channel occupancy time distribution in a cellular radio system. *IEEE Transactions on Vehicular Technology*, 36(3):98–99, 1987.

[15] R. Jain, A. Puri, and R. Sengupta. Geographical routing using partial information for wireless ad hoc networks. *IEEE Personal Comm. Magazine*, 8(1):48–57, 2001.

[16] G. Kao, T. Fevens, and J. Opatrny. Position-Based Routing on 3D Geometric Graphs in Mobile Ad Hoc Networks. In *Proc. of 17th CCCG*, pages 88–91, August 2005.

[17] G. Kao, T. Fevens, and J. Opatrny. 3-D localized Position-Based Routing with Nearly certain Delivery in Mobile Ad Hoc Networks. In *Proc. of 2nd ISWPC*, San Juan, Puerto Rico, February 2007.

[18] B. Karp and H. Kung. GPSR: Greedy Perimeter Stateless Routing For Wireless Net-Works. In *Proc. of 6th MOBICOM*, pages 243–254, Boston, August 2000.

[19] Y. Ko and N. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. *Wireless Networks*, 6(4):307–321, July 2000.

[20] E. Kranakis, D. Krizanc, L. Narayanan, A. Patnaik, and S. Shende. Routing with uncertainty in the position of the destination. In *Proc. of WiMob*, pages 9–16, 2006.

[21] E. Kranakis, H. Singh, and J. Urrutia. Compass Routing On Geometric Networks. In *Proc. of 11th CCCG*, pages 51–54, Vancouver, August 1999.

[22] X. Li. Applications of computational geometry in wireless ad hoc networks. In X. Cheng, X. Huang, and D. Du, editors, *Ad Hoc Wireless Networking*. Kluwer, 2003.

[23] M. Mauve, J. Widmer, and H. Hartenstein. A Survey of Position-Based Routing in Mobile Ad-Hoc Networks. *IEEE Network Magazine*, 15(6):30–39, November 2001.

[24] D. Pompili and T. Melodia. Three-dimensional routing in underwater acoustic sensor networks. In *Proc. of PE-WASUN*, Montréal, Canada, 2005.

[25] E. Royer, P. Melliar-Smith, and L. Moser. An analysis of the optimum node density for ad hoc mobile networks. In *Proc. of ICC*, pages 857–861, Helsinki, Finland, June 2001.

[26] I. Stojmenovic and S. Datta. Power and cost aware localized routing with guaranteed delivery in wireless networks. In *Proc. of 7th ISCC*, pages 31–36, July 2002.

[27] I. Stojmenovic and X. Lin. Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 12(10):1023–1032, 2001.

[28] G. Toussaint. The relative neighbourhood graph of a finite planar set. *Pattern Recognition*, 12(4):261–268, 1980.

[29] K. Yamazaki and K. Sezaki. The Proposal of Geographical Routing Protocols for Location-Aware Services. *Electronics and Communications in Japan*, 87(4):26–34, April 2004.