Trajectory-Based Forwarding Mechanisms for Ad-Hoc Sensor Networks

Murat Yuksel, Ritesh Pradhan, Shivkumar Kalyanaraman
Rensselaer Polytechnic Institute, Troy, NY
yuksem@rpi.edu, pradhr@ecse.rpi.edu, shivkuma@ecse.rpi.edu

Abstract—**Routing in ad-hoc sensor networks is a complicated task because of many reasons. The nodes are low powered and they cannot maintain routing tables large enough for well-known routing protocols. Because of that, greedy forwarding at intermediate nodes is desirable in ad-hoc networks. Also, for traffic engineering, multipath capabilities are important. So, it is desirable to define routes at the source like in Source Based Routing (SBR) while performing greedy forwarding at intermediate nodes.**

We investigate Trajectory-Based Routing (TBR) which was proposed as a middle-ground between SBR and greedy forwarding techniques. In TBR, source encodes trajectory to be traversed and embeds it into each packet. Upon the arrival of each packet, intermediate nodes decode the trajectory and employ greedy forwarding techniques such that the packet follows its trajectory as much as possible.

In this paper, we provide techniques to efficiently forward packets along a trajectory defined as a parametric curve. We use the well-known Bezier parametric curve for encoding trajectories into packets at source. Based on this trajectory encoding, we develop and evaluate various greedy forwarding algorithms. We also investigate various issues regarding implementation of TBR.

Keywords— **Ad-hoc Sensor Networks, Trajectory-Based Routing, Greedy Forwarding**

I. Introduction

Ad-hoc sensor networks have their own characteristics which lead to significant amount of research in the area. Particularly, routing in ad-hoc sensor networks is a complicated task because of many reasons. For example, nodes are low powered and they cannot maintain routing tables large enough for well-known routing protocols such as Link-State Routing [1]. This is known as *stateless routing* [2], since nodes do not maintain routing tables representing network state. Moreover, nodes are mobile which makes it harder to converge for typical proactive routing protocols.

So, because of its stateless nature, greedy forwarding (such as GPSR [2] and Cartesian Routing (CR) [3]) of packets at intermediate nodes is desirable in ad-hoc networks. Also, for traffic engineering, multipath capabilities are important. However, it is not possible to employ well-known multipath routing techniques (e.g. MPLS [4], or others [5]) in ad-hoc networks, because nodes are mobile. So, it is desirable to define routes at the source like in Source Based Routing (SBR) [6]. Niculescu and Nath [7] proposed Trajectory-Based Routing (TBR) as a middle-ground between SBR and greedy forwarding techniques. In TBR, source encodes trajectory to traverse and embeds it into each packet. Upon the arrival of each packet, intermediate nodes employ greedy forwarding techniques such that the packet follows its trajectory as much as possible. This way, routing becomes source-based while there is no need for routing tables for forwarding at intermediate nodes.

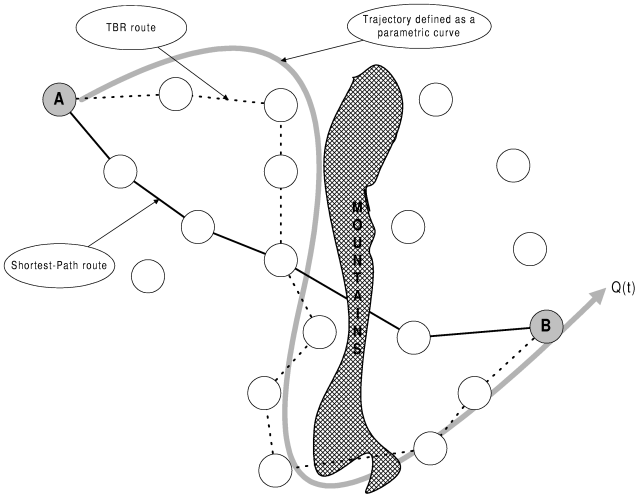


Fig. 1. An example for using TBR in an application: The application collects photos of the "west of mountains", which causes best route to be different than traditional shortest-path routing.

Furthermore as another motivation for TBR, there is a new trend toward *application-driven networking* [8] where applications can communicate with network and customize network behavior based on their own requirements. For example, consider an image processing application which collects pictures taken at different nodes in the network and merges them into a single picture of a scene. Consider the example network in Figure 1. Assume

This work is sponsored by NSF under contract number ANI9819112.

that the application is running at nodes A and B, and wants to create a big picture for the west of mountains. Observe that traditional shortest-path routing is not suitable for this type of application since the shortest path from A to B traverses nodes that are far from the west of mountains. A more suitable routing for this application is to route such that traffic of this application traverses nodes that are close to the trajectory defined as *the west of mountains.* This trajectory is also drawn as a parametric curve in the Figure 1. So, TBR is promising for such applications, examples of which can be extended.

In [7], Niculescu and Nath described basic features of TBR along with a Local Positioning System (LPS) which motivates TBR's implementation. Since it has a greedy forwarding mechanism, TBR needs support for positioning of wireless network nodes. Since GPS [9] is an already deployed positioning system, positioning is not a major issue for TBR. Also, for consideration of nodes unable to support GPS, Niculescu and Nath developed a positioning protocol LPS which enables positioning of non-GPS nodes with local information. So, in this paper, we assumed that the nodes have a knowledge of their positions with respect to a mutually known coordinate system. This assumption is reasonable as the use of GPS as well as other positioning tools are becoming more popular [10], [11], [12], [13], [14], [15], [16], [17].

In TBR, one important issue to explore is how to efficiently forward packets along a defined parametric curve $Q(t)$. Niculescu and Nath experimented with simple parametric curves such as sine curve, and left the question of how to encode various trajectories into packets as a parametric curve. In this paper, we propose an effective method of encoding trajectories into packets at source. Given this trajectory encoding techniques at source, we present various mechanisms to perform forwarding at intermediate nodes.

For trajectory encoding, we propose to use Bezier curves [18] which give a lot of flexibility in the greedy forwarding of TBR while it is possible to define a broad range of curves with them. Later in Section II, we will describe details of using Bezier curves for TBR.

The rest of paper is organized as follows: First, in Section II we describe details of Bezier curves and how to use them for trajectory encoding in TBR. Next in Section III, we propose various greedy algorithms for packet forwarding in TBR with Bezier curves. In Section IV, we present ns-2 simulations of the forwarding algorithms and evaluate their performance. Finally, in Section V we summarize the work.

II. Using Bezier Curves for TBR

In this section, we will discuss the basics of Bezier curves used for TBR. Bezier curves are special types of curves that are used in the area of graphics for representing letters in special purpose fonts. These curves are defined by a number of points - *source*, *destination*, and some *control points.* Depending on the number of control points, they are named accordingly. For instance, a Bezier curve defined by 1 *control* point is called as **quadratic Bezier curve**, while the one which is defined by 2 *control* points is known as **cubic Bezier curve**. There are other forms of Bezier curves such as **quintine Bezier curves** (3 control points), but our choice of using cubic Bezier curve was dictated by its simplicity as well as ease of computation.

A. Basics of Bezier curves

A Bezier curve $Q(t)$ is, generally, represented in its parametric form. When parameter $t = 0$, it represents the source point of the curve, while $t = 1$ represents the destination point of the curve.

A cubic Bezier curve is represented as:

$$Q(t) = \mathbf{X} = \mathbf{A}\,t^3 + \mathbf{B}\,t^2 + \mathbf{C}\,t + \mathbf{X}_0 \qquad (1)$$

where

$$\mathbf{X} = \begin{bmatrix} x \\ y \end{bmatrix}, \mathbf{A} = \begin{bmatrix} a_x \\ a_y \end{bmatrix}, \mathbf{B} = \begin{bmatrix} b_x \\ b_y \end{bmatrix},$$

$$\mathbf{C} = \begin{bmatrix} c_x \\ c_y \end{bmatrix}, \mathbf{X}_0 = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}.$$

Now, if we know the coordinates of the source (x_0, y_0), destination (x_1, y_1), and the 2 control points (x_2, y_2) and (x_3, y_3), we can calculate constants **A**, **B**, and **C** as under:

$$\mathbf{C} = 3\,(\mathbf{X}_1 - \mathbf{X}_0)$$
$$\mathbf{B} = 3\,(\mathbf{X}_2 - \mathbf{X}_1) - \mathbf{C}$$
$$\mathbf{A} = \mathbf{X}_3 - \mathbf{X}_0 - \mathbf{C} - \mathbf{B}$$

Here, $\mathbf{X}_1$, $\mathbf{X}_2$, and $\mathbf{X}_3$ are vectors similar to $\mathbf{X}$ containing the *x* and *y* coordinates of *control point-1*, *control point-2*, and destination point respectively.

Thus, from (1) we can observe that as we increase the value of the parameter t from 0 to 1, we can traverse the Bezier curve completely.

B. Closest Point on the Bezier Curve

Given a trajectory defined by a Bezier curve, the nodes can either be on the Bezier curve or could be near the

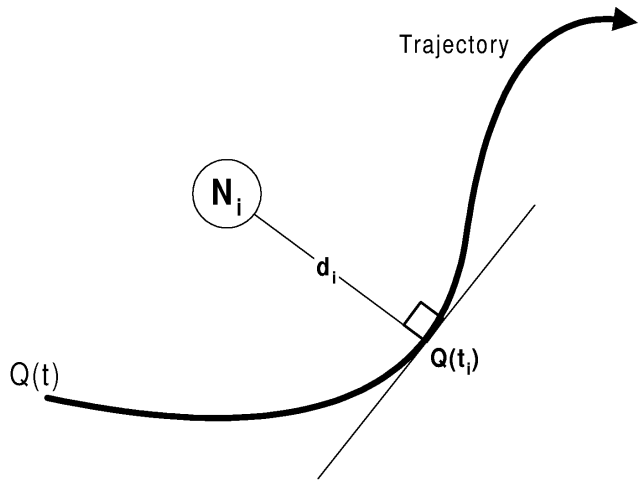


Fig. 2. A node near a trajectory defined by a Bezier curve $Q(t)$.

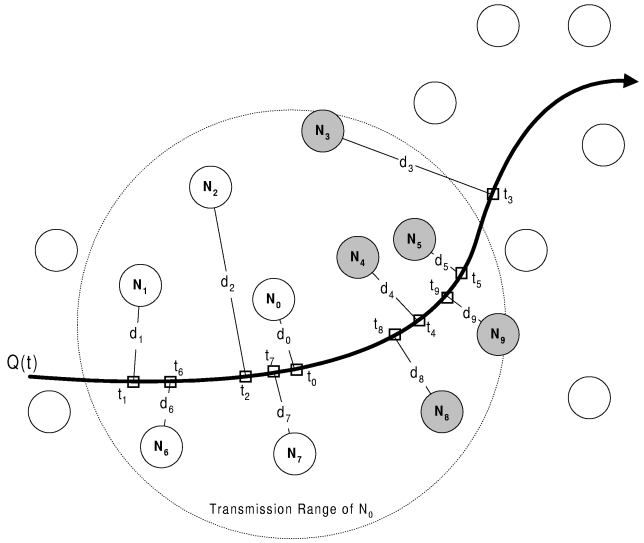


Fig. 3. Big picture of forwarding in TBR.

Bezier curve. In order to implement forwarding algorithms, for a node near the Bezier curve, we need to find where this node corresponds on the Bezier curve. This is actually the point on the curve closest to the node.

Finding the Bezier curve point closest to a node is a non-trivial task. In the Figure 2, the node does not lie on the Bezier curve. To calculate the point on the curve which is is nearest to the node, we draw a perpendicular on the tangent of the curve. Now, with $Q(t)$ in (1) being a third order polynomial and the tangent $Q'(t)$ being a second order polynomial, we get a fifth order polynomial when we have $Q(t)Q'(t) = 0$. One of roots of this equation will be the point on the Bezier curve $Q(t)$ nearest to the node [19]. Roots of a fifth degree polynomial can be computed but finding roots of the polynomial with order greater than 5 is impossible.

Given the above methodology to find the nearest point a Bezier curve, we now fix a terminology to ease writing rest of the paper. Given a Bezier curve $Q(t)$ and a node N_i as shown in Figure 2, we call the value of parameter t at the curve point closest to N_i as *residual* of N_i and represent it by t_i. The closest curve point itself is called as *residual point of* N_i, and represented by $Q(t_i)$. Finally, we call the distance between the node and $Q(t_i)$ as the *residual distance of* N_i and represent it by d_i.

III. Greedy Forwarding Algorithms for TBR

Given a neighborhood and a trajectory to follow for the packet, a node may follow different forwarding strategies depending on application and user criteria. One can define various objectives for forwarding in TBR:

- *Obey the trajectory:* There might be cases where obeying the trajectory is critical. For example, if the trajectory is passing through just near enemy area in a battlefield, then making sure that packets are obeying the trajectory and are not getting to the enemy area is important. This becomes particularly important when packets include secure information that must not reach to enemy wireless agents.
- *Reach the destination node:* As another criteria, if application generating the packets is sensitive loss of packets, then one might find it more convenient to forward the packet to the destination node if it is in the neighborhood of the forwarding node although it might be disobeying the trajectory significantly.
- *Reach quickly:* If the information being sent is delay sensitive and the similarity of route to trajectory is not of much importance, then it becomes more convenient to forward the packets such that they reach to the destination as quick as possible.

For usefulness of the forwarding strategy, the forwarding algorithm must make sure that the packet advances along the trajectory curve. In other words, a node should not forward a packet backwards along the trajectory curve. For example, in Figure 3, consider node N_0 with residual t_0. Although there are other nodes within the transmission range of N_0, the forwarding algorithm must forward packets to one of the gray nodes whose residuals are larger than t_0. We will call the set of nodes that have residuals larger than t_0 as *neighborhood*[1] *of* N_0. Within the neighborhood, selection of which node to forward packets next depends on various user and application objectives, some of which were itemized above.

[1] Note that our definition of neighborhood is different from Niculescu and Nath's definition in [7].

As another important issue, the simplicity of the forwarding algorithm is crucial for implementation purposes. Since agents are generally low-powered in wireless networks (particularly in sensor networks), computational simplicity is an important factor in terms of deployment.

In the following sub-sections, we develop algorithms for selection of next node within the neighborhood according to the above-mentioned various forwarding criteria. Note that all the following forwarding algorithms assume that the set of nodes that are composing the neighborhood is calculated. This only requires residuals to be calculated for every single node within the transmission range. Given residuals of nodes in the transmission range, one can easily construct the neighborhood of the current node (the node where the packet is currently residing) by simply comparing residuals to the residual of the current node.

A. Random

A simple algorithm is to select the next node randomly from the neighborhood. This algorithm is beneficial when computation power is of critical importance. Also, if transmission power of nodes in the network is relatively small, then this algorithm will perform fine since nodes will not have very large neighborhoods that may cause packets to be forwarded far away from the trajectory. So, the Random algorithm may be useful for wireless networks with nodes having low computational and transmission power.

B. Closest to Curve (CTC)

Another computationally simple algorithm is to select the node which is closest to the curve among the nodes in neighborhood. This algorithm is pretty straightforward to implement. Simply, calculate residual distances of each node in the neighborhood and select the one resulting in the smallest residual distance.

If obeying to the trajectory is important, then CTC is more useful. This algorithm is again useful for the cases where computational power is of critical importance. However, it may result in significant errors in forwarding such as shown in Figure 4. Since residual distance d_5 of node N_5 is smaller than residual distances all the other nodes in the neighborhood, N_0 forwards packet to N_5 which causes a significant violation of the trajectory.

C. Least Advancement on Curve (LAC)

One might need to traverse all the nodes that are along the trajectory curve. For example, if an information needs to be flooded in the network, application may want its packets to traverse as much nodes as possible. A simple algorithm is to forward to the node whose residual lies right next to the residual of the current node. Note that this algorithm is also useful for low computation powered networks.

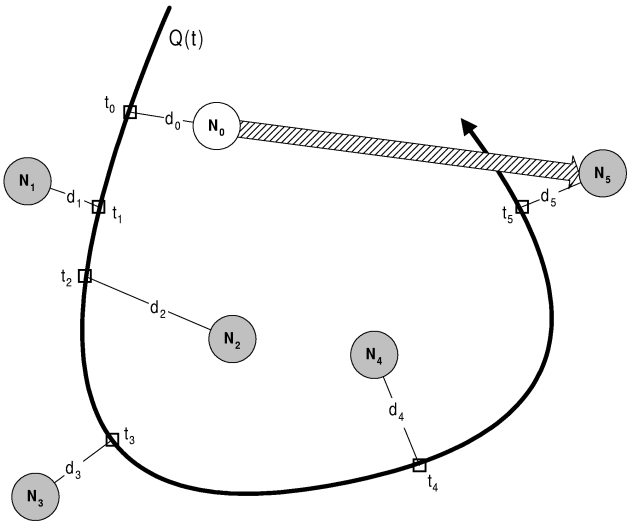


Fig. 4. Failure of CTC and MAC forwarding.

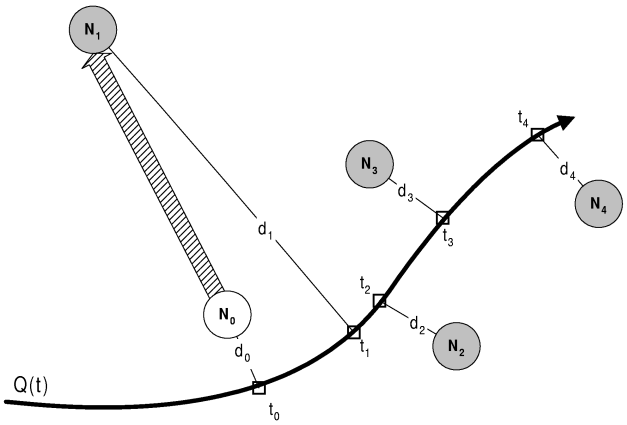


Fig. 5. Failure of LAC forwarding.

This means all the nodes that are within the transmission range will be traversed one after another according to the order of their residuals. However, again, this might result in significant errors in forwarding such as in Figure 5. Although N_1 is the farthest node from the trajectory curve, N_0 forwards packets to N_1 because t_1 is less than residuals of all the other nodes in the neighborhood of N_0.

D. Hybrid of CTC and LAC (CTC-LAC)

Another possibility is to combine CTC and LAC when one want traverse as many nodes as possible while trying to obey the trajectory curve. Combining CTC and LAC can be done in various ways depending on importance of obeying the trajectory relative to importance of traversing as many nodes as possible. We assume that obeying to the trajectory is of more importance.

A computationally simple algorithm is as follows: First,

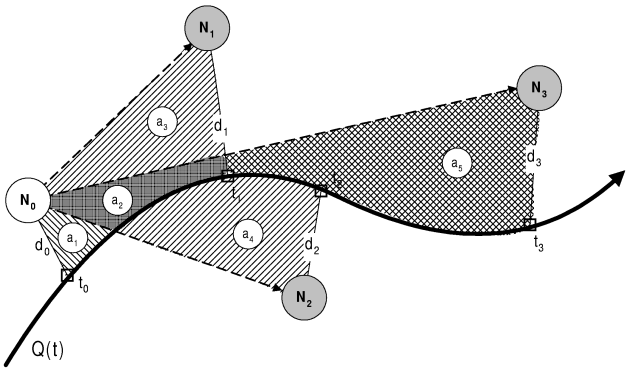


Fig. 6. Big picture of LDC forwarding.

define a tolerable residual distance D. Then, go through the neighborhood and try to find a neighbor node N_i having residual distance $d_i < D$. If there are multiple nodes satisfying the condition $d_i < D$, then select the one with smallest residual t_i. If there is no nodes satisfying the condition, then increment D with a step value $\triangle D$ and try again until a node is selected as the next node.

E. Most Advancement on Curve (MAC)

If delay is of more importance, one might want to forward the packets to the farthest node along the curve. This is again a simple algorithm to implement since just calculation of residuals will be enough in order to find out the farthest node to the current node. However, MAC forwarding may cause significant violations of trajectory as shown in Figure 4.

Similar to CTC-LAC, it is also possible to combine CTC with MAC. However, we skip developing a hybrid algorithm between CTC and MAC, since it is pretty similar to CTC-LAC.

F. Lowest Deviation from Curve (LDC)

When obeying the trajectory is very crucial, it is possible to select the next node such that the taken route deviates from the trajectory as less as possible. However, this requires extra computations. We now describe how to implement such an algorithm.

In order to obey the trajectory at most level, at a current node N_0, the best next node N_i should be selected such that the line between N_0 and N_i must have the smallest deviation from the trajectory compared to the other lines between N_0 and any other node in N_0's neighborhood. Let A_i be the area between the line N_0-N_i and the curve, i.e. the total deviation of the forwarding from the trajectory. In order to minimize the average deviation from the trajectory, the next node selection must minimize ratio of A_i by the change in residuals $t_i - t_0$, i.e. the deviation from trajectory per unit length of the curve. So for node N_0, we can write the ratio to minimize as:

$$R_i = \frac{A_i}{t_i - t_0} = \frac{Area(N_0, N_i, Q(t_0), Q(t_i))}{t_i - t_0}$$

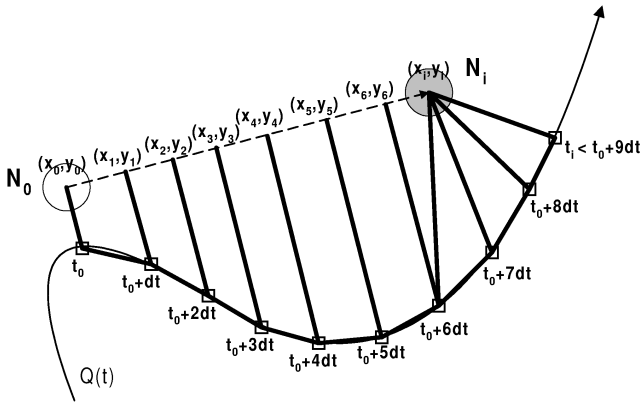
for all N_i in neighborhood of N_0. Figure 6 shows big picture of the necessary area calculations for LDC forwarding at node N_0. To illustrate an example, N_0 needs to calculate $A_1 = a_1 + a_2 + a_3$, $A_2 = a_1 + a_4$, and $A_3 = a_1 + a_2 + a_5$.

The problem is that, however, calculation of A_i requires extra computations and is not trivial. Closed-form analytical expressions for A_i are very hard to obtain. Fortunately, we can approximate A_i by numerical techniques similar to the method of Riemann sums [19] in numerical integration. We now describe how to approximate A_i.

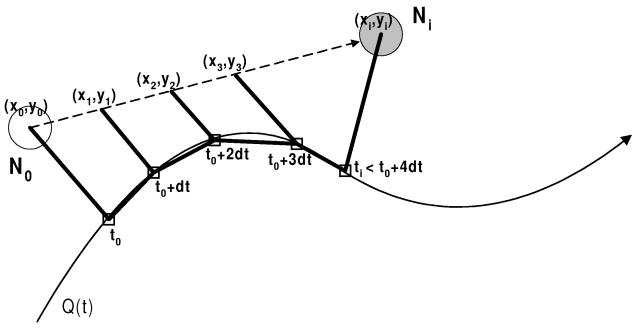
Starting from the residual t_0, we move along the curve with a fixed increase dt in the curve parameter t. At the beginning we know the points: (x_0, y_0), $Q(t_0)$. We first calculate $Q(t_0 + dt)$ and draw the line $Q(t_0)$-$Q(t_0 + dt)$. Then, we draw a line from $Q(t_0 + dt)$ toward the forwarding line (x_0, y_0)-(x_i, y_i) parallel to the line $Q(t_0)$-(x_0, y_0). Let (x_1, y_1) be the point where our new line intersects the forwarding line (x_0, y_0)-(x_i, y_i). By using the slopes of lines (x_0, y_0)-(x_i, y_i) and $Q(t_0)$-(x_0, y_0), we calculate the point (x_1, y_1). Now, we have a trapezoid between drawn by points: $Q(t_0)$, (x_0, y_0), $Q(t_0 + dt)$, and (x_1, y_1). Since we know coordinates of all the four points we can calculate the area of the trapezoid. As shown in Figures 7-a and 7-b, we, then, iterate the procedure by incrementing the residual to $t_0 + 2dt$ and generate a new trapezoid. This iteration continues until either the residual on the curve passes t_i or the intersection point on the forwarding line passes (x_i, y_i). In other words, we make n iterations if one of the two conditions is met: $t_i < t_0 + (n + 1)dt$ or $(x_n, y_n) < (x_i, y_i) < (x_{n+1}, y_{n+1})$. Depending on which condition is satisfied first, we calculate the rest of the area A_i accordingly.

Figure 7-b shows an example of the case when the former condition is satisfied first. We simply draw a quadrilateral between the four points: (x_n, y_n), $Q(t_0 + ndt)$, $Q(t_i)$, and (x_i, y_i). We can easily calculate area of this quadrilateral since coordinate of all the four points are available.

Figure 7-a shows an example of the case when the former condition is satisfied first. We first calculate the triangular area between the points: (x_n, y_n), $Q(t_0 + ndt)$, and (x_i, y_i). Then, we keep incrementing the residual until the former condition is satisfied. At each iteration we calculate the triangular area generated by drawing a line between (x_i, y_i) and the new point on the curve. In other words, at iteration $n+m$, we calculate the area of the triangle between points: (x_i, y_i), $Q(t_0 + (n + m - 1)dt)$, and



(a) Case I: $(x_n, y_n) < (x_i, y_i) < (x_{n+1}, y_{n+1})$



(b) Case II: $t_i < t_0 + (n+1)dt$

Fig. 7. Calculation of area between the Bezier trajectory and the forwarding line.

$Q(t_0 + (n+m)dt)$. Finally, when the former condition is met we simply calculate the triangular area between the points (x_i, y_i), $Q(t_i)$, and the last point on the curve (i.e. $Q(t_0 + (n+k)dt)$ if the condition was met at iteration $n+k+1$).

The approximation to A_i is simply accumulation of the areas of the small pieces that were generated during the procedure above. Of course, approximation will perform better when the residual increment dt is smaller.

LDC is expected to perform optimally if *obeying the trajectory* is the only and the most important objective in TBR. Given the *local* information only, it provides the best way of selecting the next node whom packets to be forwarded. In order to optimize the overall route taken by packets of a trajectory, better techniques can be developed when non-local information is available to forwarding nodes.

When computational simplicity is important one might want to use CTC instead of LDC with the trade-off that it may cause significant errors such as the one shown in Figure 4. An interesting observation is that CTC performance will be very close to LDC performance in dense networks. So, in heavily dense networks CTC may be a better choice than LDC.

IV. Simulations

In order to evaluate the forwarding algorithms we developed for trajectory-based routing with Bezier curves, we ran extensive simulations in ns-2. We particularly looked at two metrics: average deviation from trajectory and average path length.

We simulated the forwarding algorithms for two different trajectories: circular and zig-zag. Trajectories are shown in Figure 8 over a scenario with 75 nodes. We varied number of nodes in the simulation from 20 to 300.

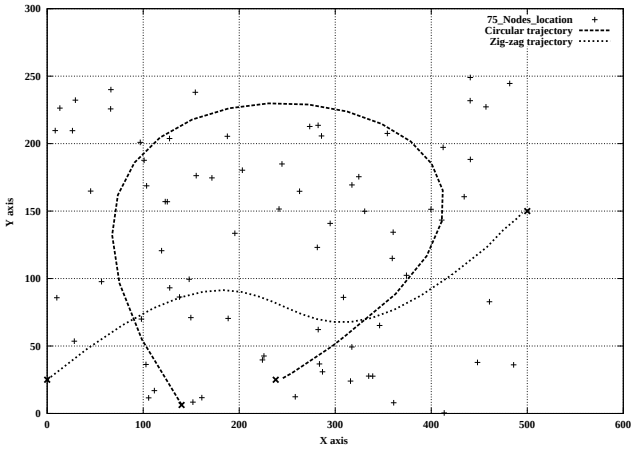


Fig. 8. Experimental trajectories seen in a scenario with 75 nodes.

Each node is a wireless node with an omnidirectional antenna. Transmission range of antennas is 5m in radius and the antennas are placed 0.9m higher than XY-plane.

The wireless nodes are exchanging beacons with an interval of 10s. Each node maintains a neighbor table, each entity of which expires if no new beacon has been received within the last 110s.

In our simulations, nodes are randomly distributed over a rectangular area 250mX500m. We picked a source-destination pair such that source is close to the starting point of trajectory and the destination node is close to the ending point of trajectory. The source generates CBR traffic with average packet size of 0.5KB. Total simulation time is 1000s.

Figures 9-a and 9-b show average deviation of packets's routes from the ideal trajectory, for the case of circular and zig-zag trajectories respectively. We observe that LDC is outperforming the other forwarding algorithms in the case of circular trajectory. Sometimes, CTC outperforms LDC

which explains the fact that LDC is making local optimization without considering next hop's choice. This causes CTC to win sometimes. In both trajectories, we see that LDC and CTC is converging to each other as density of nodes increases. However, we observe CTC failure (as explained in Figure 4) in some cases such as when number of nodes is 250 in circular trajectory.

Also, LAC and MAC performs worse than the others in general, which is caused by LAC's and MAC's ignorance on obeying to trajectory. As expected, CTC-LAC performs in between CTC and LAC. Nicely, we observe that Random forwarding performs average compared to other forwarding algorithms.

Figures 10-a and 10-b show average path length traversed by packets normalized to the length of the ideal trajectory, for the case of circular and zig-zag trajectories respectively. We can observer that, as expected, LAC performs worst in terms of path length. MAC outperforms all the other for the circular trajectory, however it is beaten by CTC and CTC-LAC for the zig-zag trajectory. That difference becomes more evident as density of nodes increases.

For the circular trajectory, normalized path length is approximately 1 for LDC, which also shows that LDC is the one that obeys the trajectory most. However, for zig-zag trajectory, LDC becomes larger than 1 as density of nodes increases. This means LDC is best for moderately populated ad-hoc networks. This discourages use of LDC for very dense networks since its computational overhead is more for denser networks (as number of neighbors will increase too).

Also, Random again performs average compared to the others in terms of path length. So, an interesting finding is that Random forwarding is good in order to achieve an average performance while avoiding a lot of computational overhead of more complex forwarding mechanisms.

Since we kept mobility close to zero, we did not sketch the probability of reaching destination. For Random, probability reaching destination was more than 80%. For all the others, it was more than 95%.

V. Summary

In this paper, we studied Trajectory-Based Routing(TBR) for stateless routing in ad-hoc sensor networks. We proposed using Bezier curves for defining trajectories in TBR. Various shapes for routes can be defined by using Bezier curves. We particularly developed several forwarding algorithms based on trajectories defined by Bezier curves.

We proposed an optimal forwarding algorithm, Least Deviation from Curve(LDC), that obeys to trajectories the most. We ran extensive simulations in order to evaluate the forwarding algorithms. We found that LDC is good for moderately populated ad-hoc networks. Interestingly, we also found that Random forwarding performs average while avoiding significant computational overhead.

Several issues remain to be investigated such as effect of mobility patterns, traffic patterns. Also, future work includes studying methods for increasing resilience (i.e. probabiliy of reaching to destination) for different forwarding algorithms.

If we consider applications such as traversing a river or eastern face of a mountain, these applications will require consideration of curves which could be represented by using much more number of control points than two. Such a curve will be very difficult to encode in the packet header as then we will have to encode each and every control point which would make the header large. Also, computing such a Bezier curve is extremely difficult during the time of greedy forwarding. So, it is necessary to study methods of scaling packet header, such that packet header size stays relatively small even though the trajectory is very long and complex.

Finally, as another open issue, answering the question of how to route the packets to destination when the destination and the source are mobile, which is generally the case in ad-hoc networks.

References

[1] R. Perlman, *Interconnections: Bridges and routers*, Addison-Wesley, 1992.

[2] B. Karp and H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *Proceedings of ACM MOBICOM*, 2000.

[3] G. Finn, "Routing and addressing problems in large metropolitan-scale networks," Tech. Rep., University of Southern California, March 1987.

[4] E. Rosen, A. Viswananthan, and R. Callon, "Multiprotocol label switching architecture," *IETF RFC 3031*, February 2001.

[5] D. Ganesan, R. Govindhan, S. Shenker, and D. Estrin, "Highly resilient, energy efficient multipath routing in wireless sensor networks," *Mobile Computing and Communications Review (MC2R)*, vol. 1, no. 2, 2002.

[6] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad-hoc wireless networks," *Mobile Computing*, vol. 353, 1996.

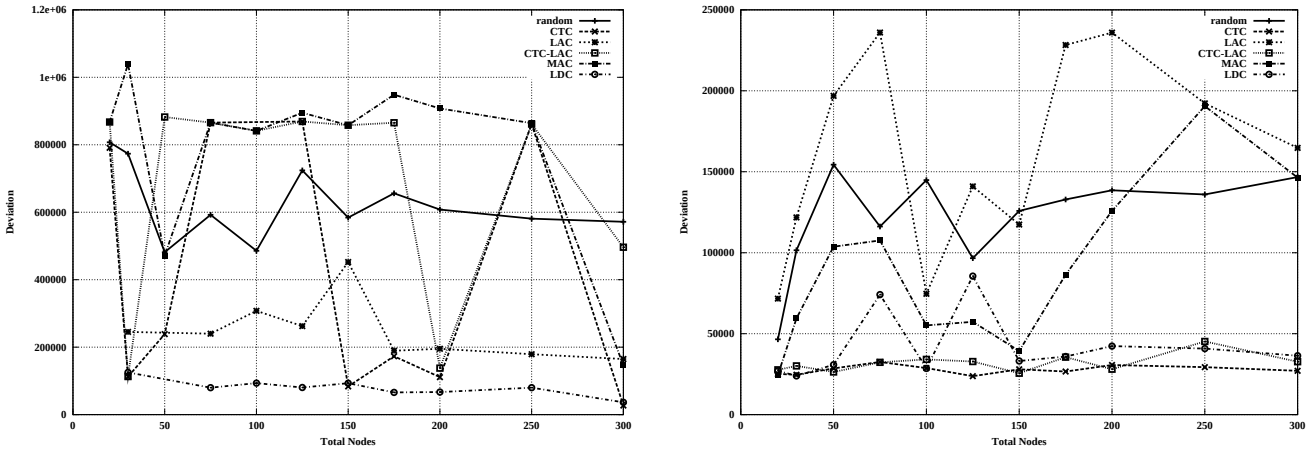
[7] D. Niculescu and B. Nath, "Trajectory based forwarding and its applications," Tech. Rep. DCS-TR-488, Rutgers University, June 2002.

[8] J. Follows and D. Straeten, *Application-Driven Networking: Concepts and Architecture for Policy-Based Systems*, IBM Red Book, 1999.

[9] B. Parkinson et al., *Global Positioning System: Theory and Application*, vol. 163, Progress in Astronautics and Aeronautics, 1996.

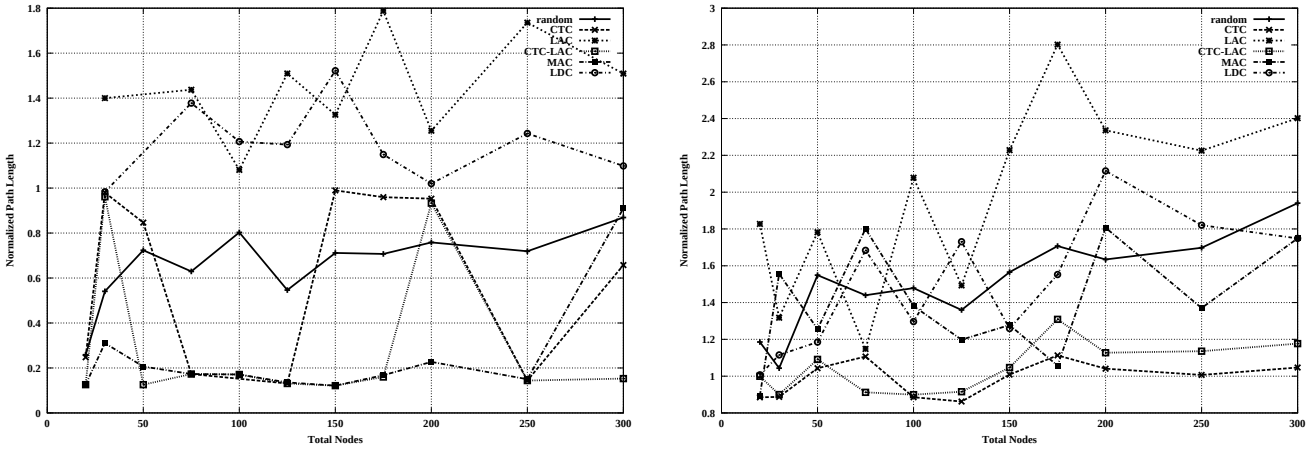
[10] D. Niculescu and B. Nath, "Ad-hoc positioning system (aps)," in *Proceedings of GLOBECOM*, 2001.

[11] J. C. Navas and T. Imielinski, "Geographic addressing and routing," in *Proceedings of ACM MOBICOM*, 1997.



(a) Circular topology (b) Zig-zag topology

Fig. 9. Average deviation from trajectory in simulation experiments.



(a) Circular topology (b) Zig-zag topology

Fig. 10. Path length normalized to trajectory length in simulation experiments.

[12] Y.-B. Ko and N. H. Vaidya, "Location-aided routing (lar) in mobile and ad-hoc networks," in *Proceedings of ACM MOBICOM*, 1998.

[13] J. Li et al., "A scalable location service for geographic ad-hoc routing," in *Proceedings of ACM MOBICOM*, 2000.

[14] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in *Proceedings of ACM MOBICOM*, 2001.

[15] P. Bahl and V. N. Padmanabhan, "RADAR: an in-building RF-based user location and tracking system," in *Proceedings of Conference on Computer Communications (INFOCOM)*, 2000.

[16] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low cost outdoor localication for very small devices," *IEEE Personal Communications Magazine, Special Issue on Smart Spaces and Environments*, October 2000.

[17] S. Capkun, M. Hamdi, and J. P. Hubaux, "GPS-free positioning in mobile ad-hoc networks," in *Proceedings of Hawaii International Conference on System Sciences (HICSS)*, 2001.

[18] P. J. Schneider and D. H. Eberly, *Geometric Tools for Computer Graphics*, 2002.

[19] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, 1992.