Available online at www.sciencedirect.com
SCIENCE DIRECT®
Ad Hoc Networks xxx (2004) xxx–xxx
Ad Hoc Networks
www.elsevier.com/locate/adhoc

An implementation framework for trajectory-based routing in ad hoc networks

Murat Yuksel *, Ritesh Pradhan [1], Shivkumar Kalyanaraman

ECSE Department, Rensselaer Polytechnic Institute, JEC-6049, 110 8th Street, Troy, NY 12180, USA

Received 5 November 2003; accepted 14 April 2004

Abstract

Routing in ad hoc networks is a complicated task because of many reasons. The nodes are low-memory, 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, multi-path 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 address various issues regarding implementation of TBR. We also 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

Keywords: Trajectory-based routing; Greedy forwarding; Stateless networking

1. Introduction

Ad hoc networks have their own characteristics which lead to significant amount of research in the area. Particularly, routing in ad hoc networks is a complicated task because of many reasons. For example, nodes are generally low in memory and

* Corresponding author. Tel.: +1-518-276-6823; fax: +1-925-888-2167.
E-mail addresses: yuksem@ecse.rpi.edu (M. Yuksel), rspradhan@alum.rpi.edu (R. Pradhan), shivkuma@ecse.rpi.edu (S. Kalyanaraman).
[1] R. Pradhan is now with GE Power Systems, Atlanta, GA.

1570-8705/$ - see front matter © 2004 Published by Elsevier B.V.
doi:10.1016/j.adhoc.2004.04.010

power, and hence they cannot maintain routing tables large enough for well-known link-state or distance-vector routing protocols. This is known as *stateless routing* [1], 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 (e.g. FACE [2], GPSR [1] and Cartesian routing (CR) [3]) of packets at intermediate nodes is desirable in ad hoc networks. Also, for traffic engineering, multi-path capabilities (e.g. source based routing (SBR) [4]) are desirable. However, it is not possible to employ well-known multi-path routing techniques (e.g. MPLS [5], or others [6]) in ad hoc, particularly mobile, networks. Niculescu and Nath [7,8] 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.

Furthermore as another motivation for TBR, there is a new trend toward *application-driven networking* [9], particularly in sensor networks. In this new networking paradigm, 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 3D picture of a scene. Consider the example network in Fig. 1. Assume that the application is running at nodes A and B, and wants to create a large picture that captures 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 this application's traffic traverses nodes that are close to the trajectory defined as *the west of mountains*. This trajectory is also drawn as a parametric curve in Fig. 1. So, TBR is promising for such applications,

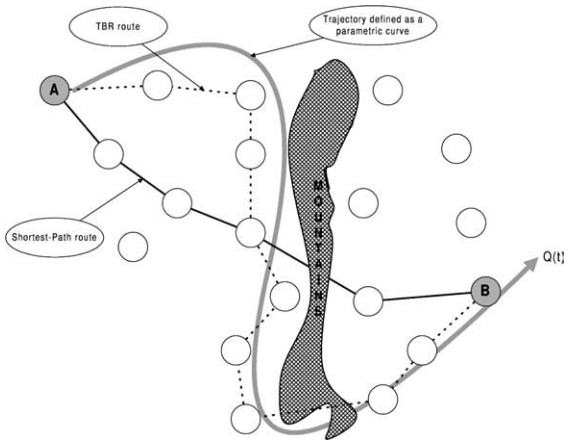

Fig. 1. An example for using TBR in an application: the application collects photos of the "west of mountains", which causes best route to be different than traditional shortest-path routing.

examples of which can be extended, e.g. collecting measurements from sensors deployed over a river, routing without crossing into an undesired area.

In [7,8], Niculescu and Nath described basic features of TBR along with a local positioning system (LPS). Since it has a greedy forwarding mechanism, TBR needs support for positioning of wireless nodes. As a solution to this problem, various positioning systems such as GPS [10] can be used. However, GPS requires high power availability which is difficult and costly in implementation of low-power ad hoc networks. For this purpose, GPS-free positioning techniques [11,12] as well as Nath and Niculescu's LPS can be used to enable TBR's implementation at low-power nodes without GPS support. So, in this paper, we assumed that 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 [13–17,11,12].

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.

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. We also describe a protocol for implementing longer and more complex trajectories as a concatenation of Bezier curves. Given this trajectory encoding technique at source, we present various mechanisms to perform forwarding at intermediate nodes.

Contributions of this paper can be listed as follows:

- Methodology for encoding and decoding trajectories by using cubic Bezier curves.
- Implementation of TBR using Bezier curves, which gives enough flexibility to define a large range of trajectories (e.g. zig-zag, circular).
- An implementation protocol of TBR for longer and more complex trajectories, virtually without any limit on the length and complexity of the trajectory.
- Implementation of the forwarding methodology, lowest deviation from curve (LDC), that performs optimally in terms of obeying the trajectory in forwarding.
- Evaluation of the forwarding technique LDC as well as several other intuitive forwarding techniques.
- Packet-based ns-2 simulation of these new implementation and forwarding techniques.

The rest of paper is organized as follows: First, in Section 2 we describe details of Bezier curves and how to use them for trajectory encoding in TBR. In Section 3 we briefly describe ways of scaling packet header for trajectory encoding with Bezier curves. Next in Section 4, we propose various greedy algorithms for packet forwarding in TBR with Bezier curves. In Section 5, we present ns-2 simulations of the forwarding algorithms and evaluate their performance. Finally, in Section 6 we summarize the work.

2. Using Bezier curves 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 one *control* point is called as *quadratic Bezier curve*, while the one which is defined by two *control* points is known as *cubic Bezier curve*. More details about basic calculations for Bezier curves can be found in [18].

There are other forms of Bezier curves such as *quintine Bezier curves* (three control points), but our choice of using cubic Bezier curve was dictated by its simplicity as well as ease of computation.

2.1. Forwarding along a cubic Bezier curve

Shape of a Bezier curve is dependent on the locations of the control points. A sample cubic Bezier curve is shown in Fig. 2. It can also be represented in its parametric form, $Q(t)$. When parameter $t=0$, it represents the source point of the curve, while $t=1$ represents the destination point of the curve.

More specifically, a cubic Bezier curve is represented algebraically as

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

where

$$\mathbf{X} = \begin{bmatrix} x \\ y \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} a_x \\ a_y \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} b_x \\ b_y \end{bmatrix},$$
$$\mathbf{C} = \begin{bmatrix} c_x \\ c_y \end{bmatrix}, \quad \mathbf{X}_0 = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}.$$

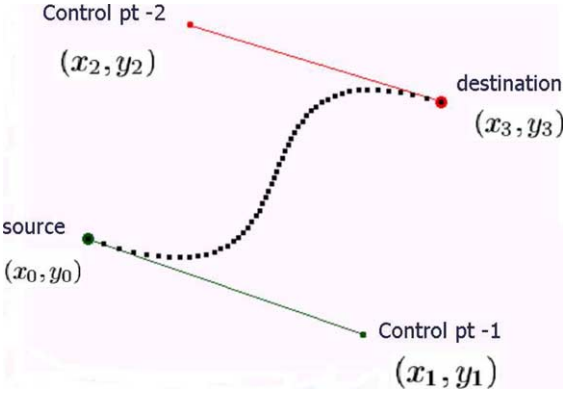

Fig. 2. A cubic Bezier curve.

For the third order polynomial shown in (1), the coefficients **A**, **B**, **C** are unique if the following equation system is satisfied: [2]

$$\begin{aligned}\mathbf{C} &= 3(\mathbf{X}_1 - \mathbf{X}_0),\\ \mathbf{B} &= 3(\mathbf{X}_2 - \mathbf{X}_1) - \mathbf{C},\\ \mathbf{A} &= \mathbf{X}_3 - \mathbf{X}_0 - \mathbf{C} - \mathbf{B}.\end{aligned} \tag{2}$$

Here, $\mathbf{X}_0$, $\mathbf{X}_1$, $\mathbf{X}_2$, and $\mathbf{X}_3$ are vectors similar to $\mathbf{X}$ containing the x and y coordinates of *source point*, *control point-1*, *control point-2*, and *destination point* respectively.

So, given the coordinates of the source (x_0, y_0), destination (x_3, y_3), and the two control points (x_1, y_1) and (x_2, y_2), one can calculate constants **A**, **B**, and **C** from the equation system (2), thereby recovering the complete Bezier curve.

Our idea is to use this mathematics to encode the complete trajectory into each packet, by putting the coordinates of source, destination, and the two control points into packet header. Then, solve the equation system in (2) to decode the complete trajectory at any intermediate node. Alternatively, putting the coordinate of source point and the three constant vectors **A**, **B**, and **C** into packet header will also work.

2.2. 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 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 Fig. 3, the node does not lie on the Bezier curve. To calculate the point on the curve which is nearest to the node, we draw a perpendicular on the tangent of the curve. Now, with $Q(t)$ 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 root 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 five is not known.

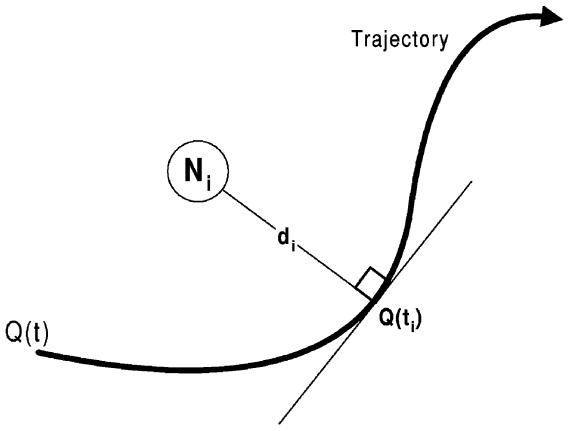

Fig. 3. A node near a trajectory defined by a Bezier curve $Q(t)$.

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 Fig. 3, 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.

3. Long or more complex trajectories

If we consider applications such as traversing a river or capturing eastern face of a mountain into a picture, 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, because we will have to encode each and every control point which would make the header bulky. Also, computation for decoding such a Bezier curve is extremely difficult during the time of greedy forwarding.

As shown in Fig. 4, one way to define long trajectories is to split the trajectory into smaller pieces

[2] Please refer to [18] for proof.

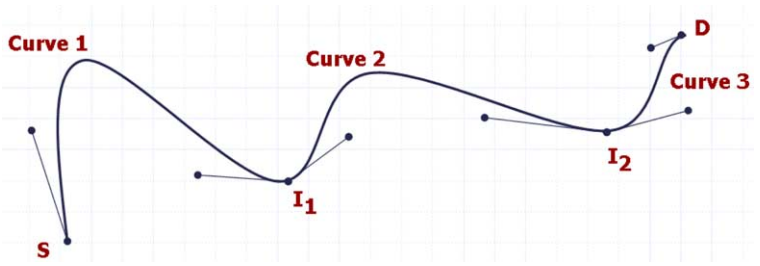

Fig. 4. A long zig-zag trajectory as a concatenation of cubic Bezier curves.

which can be represented by cubic Bezier curves (i.e. two control points). So, the complete trajectory is defined as a concatenation of a series of cubic Bezier curves. We call the concatenation points as *middle points*, shown as I_1 and I_2 in Fig. 4.

Before starting the actual transmission of data, the source can probe the ad hoc network by sending a control-plane packet which includes the whole trajectory with n control points. Upon arrival of that probe packet, an intermediate node divides the whole trajectory into equal pieces, [3] and checks whether itself is close enough (e.g. within 5 m of radius) to one of those middle points. If so, that particular node identifies itself as a *special intermediate node (SIN)* for this source-destination pair and sends an acknowledgement to the source. The source confirms SIN by replying to the acknowledgement (this is necessary to resolve contention for being SIN if there are multiple candidates close to the desired middle point). After this confirmation from the source, the SIN records the control points for the next cubic Bezier curve in the trajectory. This process will continue until all pieces of the trajectory is captured by a SIN which keeps the control points of the next cubic Bezier curve.

In this manner, there will be $n-1$ SINs, n cubic Bezier curves for a trajectory with $2n$ control points. After such a signaling protocol as described above, the source will no longer have to encode the $2n$ control points into data packets. Rather, it will just need to put two control points for the next cubic Bezier curve on the trajectory, since the next SIN will be putting the control points necessary for the following piece of the trajectory.

When the nodes are mobile, SINs can move from their original locations and may no longer be close to the trajectory. One quick solution is to send probe packets frequently throughout the data transmission. This way, SINs will be reassigned if the previous ones got away from the trajectory.

4. 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's wireless agents.
- *Reach the destination node:* As another criteria, if application generating the packets is sensitive to 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.

[3] For a Bezier curve $Q(t)$ with n control points, these pieces are portions of the whole curve in between points $Q(t/k)$ where $k=0,\ldots,n$.

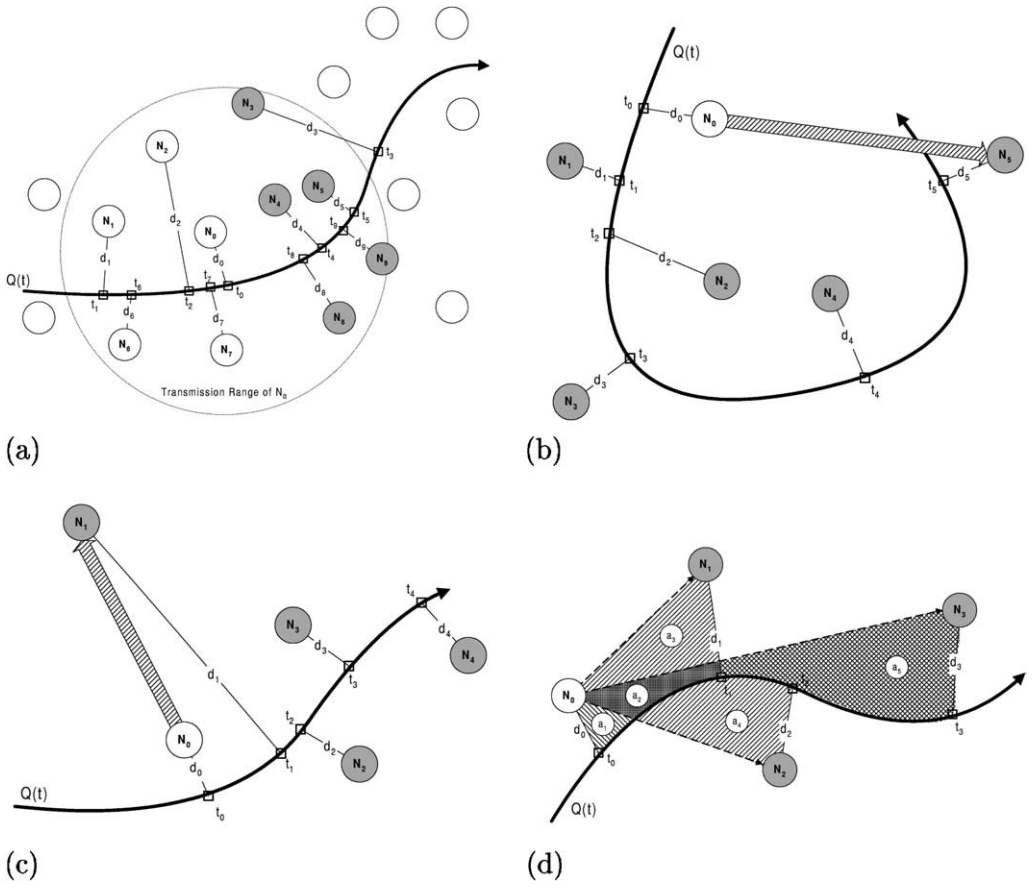

Fig. 5. Big pictures of various TBR concepts. (a) Big picture of TBR forwarding, (b) failure of CTC and MAC forwarding, (c) failure of LAC forwarding and (d) big picture of LDC forwarding.

- *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 Fig. 5a, 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* [4] 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.

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.

[4] Note that our definition of neighborhood is different from Niculescu and Nath's definition in [8].

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.

4.1. 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.

4.2. 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 Fig. 5b. 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.

4.3. 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.

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 Fig. 5c. 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.

4.4. 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, 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 ΔD and try again until a node is selected as the next node.

4.5. 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 Fig. 5b.

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.

4.6. 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

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

for all N_i in neighborhood of N_0. Fig. 5d 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.

Starting from the residual t_0, we move along the curve with a fixed increase $\mathrm{d}t$ in the curve parameter t. At the beginning we know the points: (x_0, y_0), $Q(t_0)$. We first calculate $Q(t_0 + \mathrm{d}t)$ and draw the line $Q(t_0)$–$Q(t_0 + \mathrm{d}t)$. Then, we draw a line from $Q(t_0 + \mathrm{d}t)$ 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 drawn by points: $Q(t_0)$, (x_0, y_0), $Q(t_0 + \mathrm{d}t)$, 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 Fig. 6a and b, we, then, iterate the procedure by incrementing the residual to $t_0 + 2\mathrm{d}t$ 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:

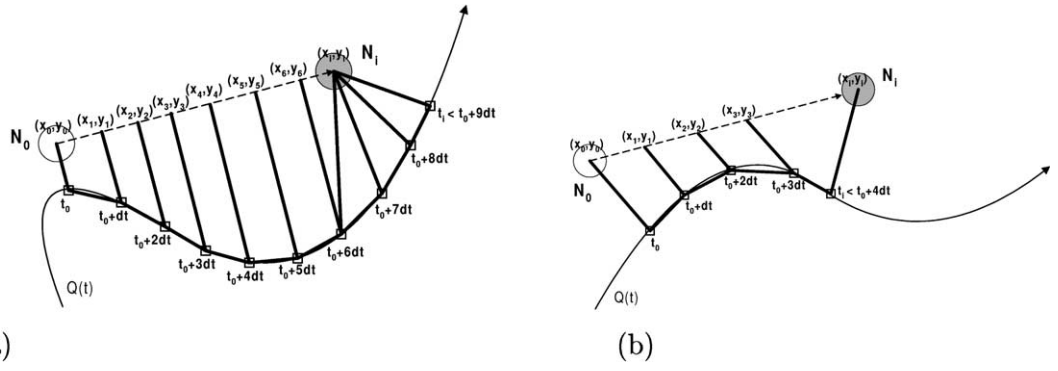

Fig. 6. Calculation of area between the Bezier trajectory and the forwarding line. (a) Case I: $(x_n, y_n) < (x_i, y_i) < (x_{n+1}, y_{n+1})$ and (b) Case II: $t_i < t_0 + (n+1)\mathrm{d}t$.

- Condition 1: $t_i < t_0 + (n+1)\mathrm{d}t$.
- Condition 2: $(x_n, y_n) < (x_i, y_i) < (x_{n+1}, y_{n+1})$.

Depending on which condition is satisfied first, we calculate the rest of the area A_i accordingly.

Fig. 6b shows an example of the case when the second condition is satisfied first. We simply draw a quadrilateral between the four points: (x_n, y_n), $Q(t_0 + n\,\mathrm{d}t)$, $Q(t_i)$, and (x_i, y_i). We can easily calculate area of this quadrilateral since coordinate of all the four points are available.

Fig. 6a shows an example of the case when the first condition is satisfied first. We first calculate the triangular area between the points: (x_n, y_n), $Q(t_0 + n\,\mathrm{d}t)$, and (x_i, y_i). Then, we keep incrementing the residual until the second 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), $Q(t_0 + (n+m-1)\,\mathrm{d}t)$, and $Q(t_0 + (n+m)\,\mathrm{d}t)$. Finally, when the second 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. $Q(t_0 + (n+k)\,\mathrm{d}t)$ 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 $\mathrm{d}t$ 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 Fig. 5b. 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.

5. Simulations

Our purpose of ns-2 simulations is twofold:

- *Evaluate the forwarding algorithms developed for TBR.*
- *Proof-of-concept for extension to longer and more complex trajectories.*

We particularly look at two metrics: average deviation from trajectory and average path length. For this paper, we do not include mobility in our simulations.

5.1. Evaluation of the forwarding algorithms

We simulated the forwarding algorithms for two different trajectories: circular and zig-zag. Trajectories are shown in Fig. 7a over a scenario with 75 nodes. We varied number of nodes in the simulation from 20 to 300. Each node is a wireless node with an omnidirectional antenna. Transmission range of antennas is 5 m in radius and the antennas are placed 0.9 m higher than XY-plane. The wireless nodes are exchanging beacons with an interval of 10 s. Each node maintains a neighbor table, each entity of which expires if no new beacon has been received within the last 110 s.

In our simulations, nodes are randomly distributed over a rectangular area 250 m$\times$500 m. 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 0.5 KB. Total simulation time is 1000 s.

Figs. 8a,b and 9a,b show average deviation of packets' 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

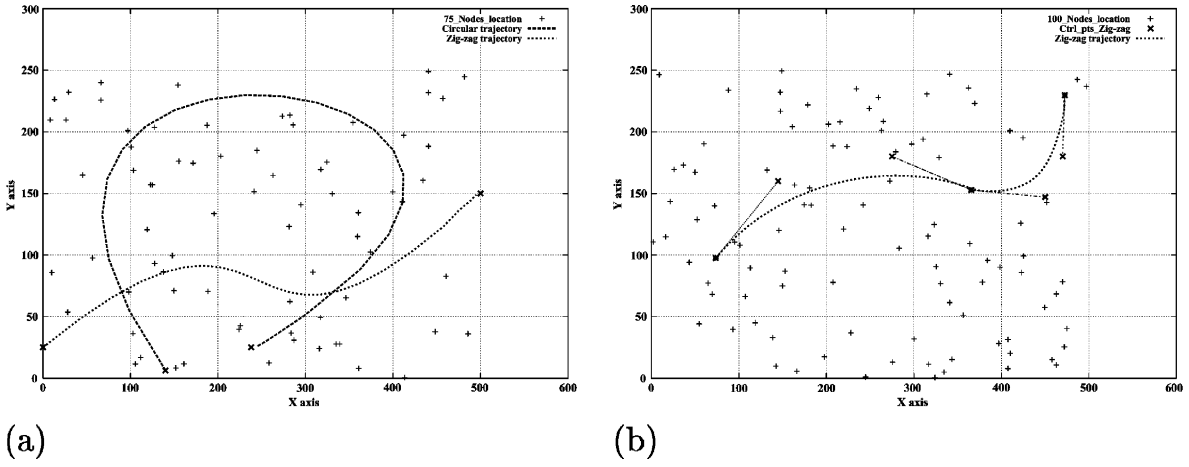

Fig. 7. Experimental trajectories: (a) Zig-zag and circular single-piece trajectories on randomly located 75 nodes. (b) The multi-piece trajectory with a single SIN and two cubic Bezier curves.

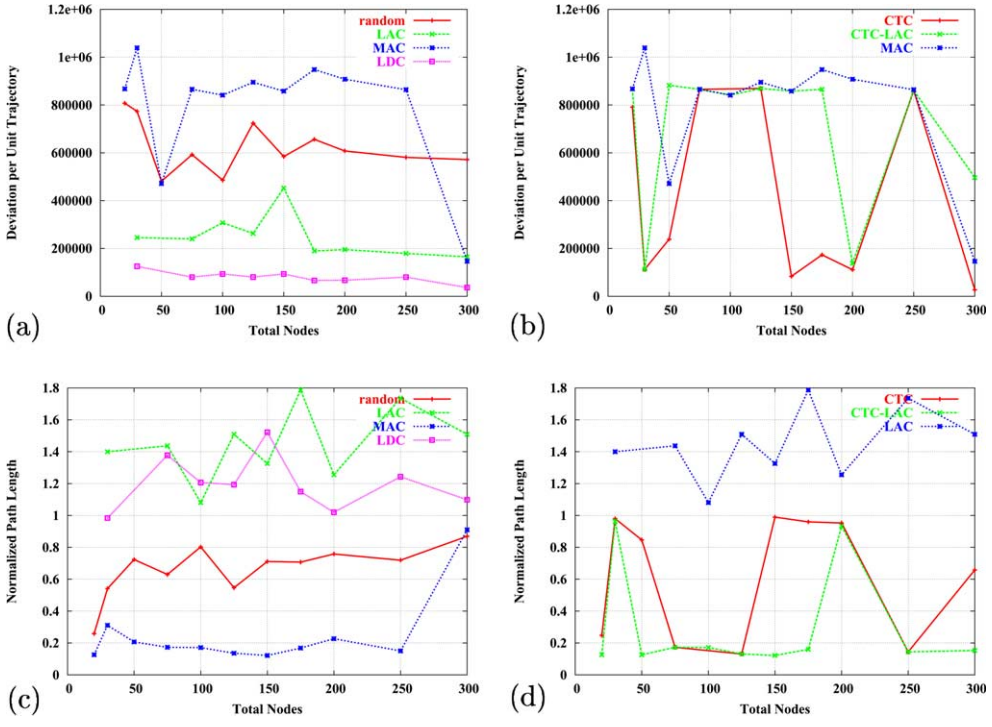

Fig. 8. Simulation results for the *circular trajectory*. (a–b) Average deviation from trajectory. (c–d) Path length normalized to trajectory length.

explained in Fig. 5b) 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

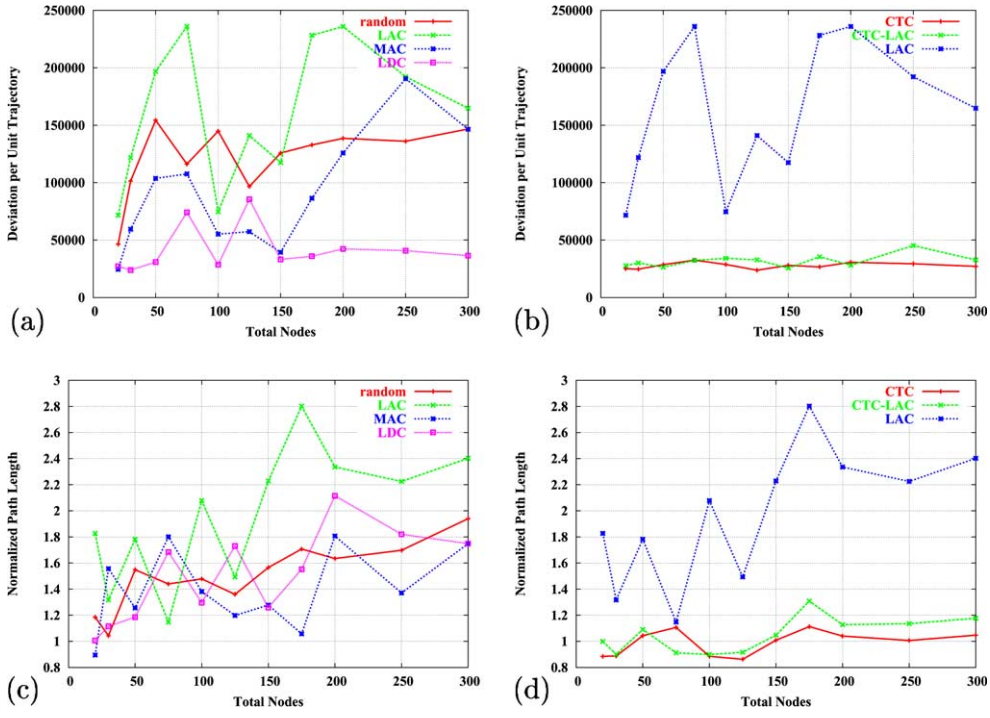

Fig. 9. Simulation results for the *zig-zag trajectory*. (a–b) Average deviation from trajectory. (c–d) Path length normalized to trajectory length.

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.

Figs. 8c,d and 9c,d 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 observe 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. For Random forwarding, probability of reaching destination was more than 80%. For all the others, it was more than 95%.

5.2. More complex trajectories

Simulation settings are the same as the one in the previous section. We simulate CTC–LAC forwarding on a zig-zag trajectory which is being defined by two cubic Bezier curves as shown in Fig. 7b. The data packets include only two control

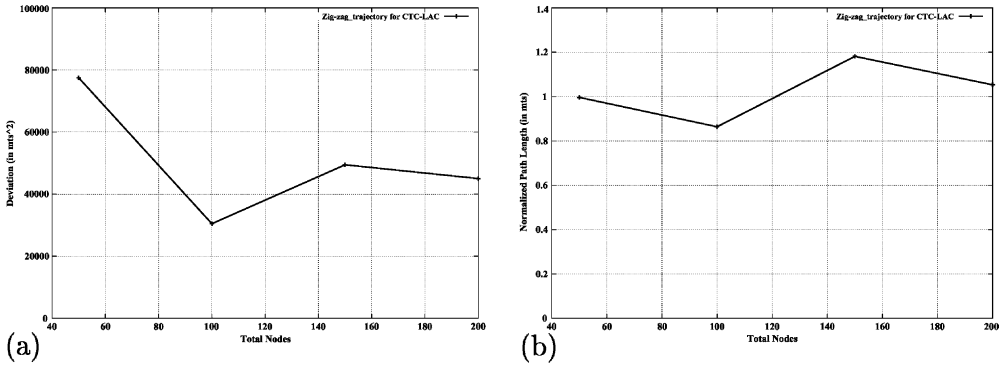

Fig. 10. Performance results for CTC–LAC forwarding on the *multi-piece trajectory*. (a) Deviation from trajectory and (b) normalized average path length.

points. There is one SIN, which splits the trajectory into two pieces.

Fig. 10a and b shows average deviation from trajectory and average path length on the zig-zag trajectory defined by two cubic Bezier pieces. We observe that average path length normalized to the ideal trajectory length is approximately 1. Also, deviation from trajectory reduces as the number of nodes increases. So, the results illustrates that packets get forwarded properly even though they include only a fraction of the control points of the complete trajectory.

6. Summary

In this paper, we studied various implementation issues of trajectory-based routing (TBR) for stateless routing in ad hoc networks. We proposed to use Bezier curves for defining trajectories in TBR. Various shapes for routes can be defined by using Bezier curves.

We particularly evaluated several forwarding algorithms based on trajectories defined by Bezier curves. We proposed an optimal forwarding algorithm, lowest 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.

By introducing signaling phase to the protocol, we also proposed a methodology for extending TBR with Bezier curves to longer and more complex trajectories which can be encoded by larger information. Our proposed method enables routing of data packets through complex trajectories, while keeping the packet header size constant. Future work will include evaluation and improvement of this method with a particular consideration given to signaling 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. probability of reaching to destination) for different forwarding algorithms.

Finally, answering the question of how to route the packets when the destination and the source are mobile, is an open issue.

References

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

[2] P. Bose, P. Morin, I. Stojmenovic, J. Urrutia, Routing with guaranteed delivery in ad hoc wireless networks, Wireless Networks 7 (6) (2001) 609–616.

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

[4] D.B. Johnson, D.A. Maltz, Dynamic source routing in ad-hoc wireless networks, in: T. Imielinski, H. Korth (Eds.), Mobile Computing, Kluwer, Dordrecht, 1996, p. 353.

[5] E. Rosen, A. Viswananthan, R. Callon, Multiprotocol label switching architecture, IETF RFC 3031, February 2001.
[6] D. Ganesan, R. Govindhan, S. Shenker, D. Estrin, Highly resilient energy efficient multipath routing in wireless sensor networks, Mobile Computing and Communications Review 1 (2) (2002).
[7] D. Niculescu, B. Nath, Routing on a curve, in: Proceedings of Workshop on Hot Topics in Networks (HOTNETS-I), 2002.
[8] D. Niculescu, B. Nath, Trajectory based forwarding and its applications, in: Proceedings of ACM MOBICOM, 2003.
[9] J. Follows, D. Straeten, Application-Driven Networking: Concepts and Architecture for Policy-Based Systems, IBM Red Book, 1999.
[10] B. Parkinson, et al., Global Positioning System: Theory and Application, Progress in Astronautics and Aeronautics, vol. 163, AIAA.
[11] N. Bulusu, J. Heidemann, D. Estrin, GPS-less low cost outdoor localization for very small devices, IEEE Personal Communications Magazine, Special Issue on Smart Spaces and Environments, October, 2000.
[12] R. lyengar, B. Sikdar, Scalable and distributed GPS free positioning for sensor networks, in: Proceedings of IEEE International Conference on Communications (ICC), 2003.
[13] D. Niculescu, B. Nath, Ad-hoc positioning system (aps), in: Proceedings of GLOBECOM, 2001.
[14] J.C. Navas, T. Imielinski, Geographic addressing and routing, in: Proceedings of ACM MOBICOM, 1997.
[15] Y.-B. Ko, N.H. Vaidya, Location-aided routing (lar) in mobile and ad-hoc networks, in: Proceedings of ACM MOBICOM, 1998.
[16] J. Li et al., A scalable location service for geographic ad-hoc routing, in: Proceedings of ACM MOBICOM, 2000.
[17] N.B. Priyantha, A. Chakraborty, H. Balakrishnan, The cricket location-support system, in: Proceedings of ACM MOBICOM, 2001.
[18] P.J. Schneider, D.H. Eberly, Geometric Tools for Computer Graphics, Morgan Kaufmann, San Francisco, CA, 2002.
[19] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes in C: The Art of Scientific Computing, Cambridge University Press, Cambridge, 1992.

Murat Yuksel is a Post-Doctoral Researcher and Lecturer at ECSE Department of Rensselaer Polytechnic Institute, Troy, NY. He received MS and PhD degrees in Computer Science from Rensselaer Polytechnic Institute in 1999 and 2002 respectively. He holds a BS degree in Computer Engineering from Ege University, Izmir, Turkey in 1996. His research is on various networking issues such as routing in wireless sensor and ad hoc networks, mobile free-space-optical networks, large-scale network simulation, network pricing, and performance analysis. He is a member of IEEE and Sigma Xi.

Ritesh Pradhan is currently working in GE Power Systems in Atlanta, GA. He is an alumnus of the Rensselaer Polytechnic Institute, where he earned his MS degrees in Electrical Engineering and Information Technology. His research interests are in the area of networking, data warehousing and Internet pricing.

Shivkumar Kalyanaraman is an Associate Professor at the Department of Electrical, Computer and Systems Engineering at Rensselaer Polytechnic Institute in Troy, NY. He received a BTech degree from the Indian Institute of Technology, Madras, India in July 1993, followed by MS and PhD degrees in Computer and Information Sciences at the Ohio State University in 1994 and 1997 respectively. His research interests are in network traffic management topics such as congestion control architectures, quality of service (QoS), high-speed wireless, free-space optical networking, network management, multicast, pricing, multimedia networking, and performance analysis. His special interest lies in developing the inter-disciplinary areas between traffic management, wireless communication, optoelectronics, control theory, economics, scalable simulation technologies, and video compression. He is an member of the ACM and IEEE.