ON THE PACKET HEADER SIZE AND NETWORK STATE TRADEOFF FOR TRAJECTORY-BASED ROUTING IN WIRELESS NETWORKS

Rajagopal Iyengar
ECSE Department
Rensselaer Polytechnic Institute, Troy, NY, 12180
Email:iyengr@rpi.edu

Murat Yuksel
Computer Science and Engineering Department
University of Nevada, Reno, NV 85997
Email:yuksem@unr.edu

ABSTRACT

This paper considers the tradeoffs between packet header size, network state and accuracy of representation in wireless networks which use trajectory based routing techniques. The models presented here are applicable in sensor networks for which trajectory based routing techniques have previously been proposed. We begin with simple discrete models to make clear the tradeoff between header size and network state. We show that the problem of accurate representation of a trajectory with the objective of minimizing the cost incurred due to header size and network state is difficult to solve optimally. Therefore, we present two heuristics to solve this problem. We note that this work leads to further interesting problems when applied to practical systems.

I. INTRODUCTION AND MOTIVATION

Geographic routing algorithms [1, 2] are among the most promising for wireless networks with highly dynamic topology and limited computational and energy resources. Appealing properties of geographic routing include that (i) they provide routing functions with very small amount of state information to be stored at wireless nodes, (ii) they allow reactive routing with small overhead. The main reason for low overhead of geographic routing schemes is that they use localization support [3, 4] to create a common coordinate system as a global-invariant among the network nodes, thereby significantly reducing state overhead.

One critical issue with geographic routing schemes is that they do not allow the source node to express an application-specific path to the routing layer. Particularly for end-to-end traffic engineering as well as application-specific purposes, capability of defining the path at the source (as in the Dynamic Source Routing (DSR)) is crucial for future wireless networks with higher quality services. To fill this gap between routing layer and higher layers, Trajectory-Based Routing (TBR) [5, 6] is a very elegant way of providing more cross-layer functionality to higher layers. TBR provides a middle-ground between source-based routing schemes and purely greedy stateless schemes like GPSR.

TBR suggests that the source encodes the trajectory into packet headers, and then the intermediate nodes forward the packets according to the trajectory decoded from their headers so as to make them traverse the source-defined trajectory as much as possible. However, for long or complex trajectories (which may not be completely encoded into packet headers) either (i) the packet header becomes very large for full encoding of the trajectory or (ii) a set of intermediate nodes (we call them as Special Intermediate Nodes (SINs)) are employed to maintain some state information so that smaller pieces of the trajectory can be encoded into smaller packet headers. The latter approach seems to be the only viable approach due to the fact that packet headers have pre-defined lengths and therefore there will certainly be some long/complex trajectories which cannot be encoded into these fixed size headers. The framework presented in [6] lays out a very practical way of implementing the latter approach where some intermediate nodes keep state for each source-destination traffic flow with a particular trajectory. Apparently, as the number of intermediate nodes maintaining state increases the needed packet header size becomes smaller, and vice versa.

In this paper we explore this important tradeoff between packet header size and the amount of network state needed to be maintained for TBR. To study this tradeoff, we formulate the problem of finding optimal representation of the complete trajectory by selecting from set of trajectory encoding methods and a number of SINs. We assume that there are certain trajectory encoding techniques (e.g. parametric curves such as Bezier and B-spline; trigonometric curves such as sine, cosine, and straight line) known by all nodes and that the nodes have hardware/software to encode/decode these basic trajectories. Then, we define the joint optimization problem of minimizing packet header size and network state where each piece of the complete trajectory can be represented by one of the available basic trajectories. To provide more flexibility in optimization, we also include an aggregate error bound on the representation of the complete trajectory. We define error measurement methods so that quality of the approximate representation can be determined easily.

A. Organization of the Paper

In Section II., a review of some of the literature on trajectory based forwarding is provided. Section III. details the optimization formulation for the trajectory approximation problem taking into account the packet header and network state costs. While the formulation is generic enough to encompass different network state and header cost models, we describe how external influences like battery power etc. can be used to model such functions. In Section IV., we present two heuristic approaches to solve the optimization problem. We conclude in V. and discuss some interesting problems which this line of work has thrown open.

B. Contributions

Broadly, we consider the problem of quantifying the tradeoff between packet header size and network state. To the best

1-4244-0330-8/06/$20.00©2006 IEEE

of the authors knowledge, this problem of studying the packet header size versus network state tradeoff when using trajectory based routing techniques has not been studied previously. The contributions of this paper are listed below:

- A generic optimization formulation for the trajectory approximation problem when we are provided with a set of curve encoding/decoding methods used to approximate portions of the trajectory.
- We show that the generic optimization problem is in general difficult to solve (it is NP-Complete) and describe heuristics to solve the problem.

II. Related Work

The approximation of curves has been a well-studied problem in the field of computational geometry. Several heuristics were proposed for approximation of digital or polygonal curves, for example in [7]. Even though this presents significant body of research in curve approximation, the focus has been on the minimizing compression costs (i.e. minimizing the number of line segments needed to satisfy an error bound [8]). The work in this paper adds a dimension to this traditional curve compression problem by including the cost of "packet header length" which is dependent on the choice of each piece in approximation. Notice that, in our problem formulation, the "piece" is not necessarily a line, rather it can be a curve which means "cost" in terms of packet header length. There has also been significant body of research in approximation of parametric curves for example [9], which mainly focus on fast and efficient compression of curves.

A very similar tradeoff to the one we study here was studied within the context of traffic engineering in wired networks [10] using Multi-Path Label Switching. Gupta et al. [11] identified and studied the tradeoff between MPLS stack depth and label sizes. Label sizes affect the amount of state that needs to be stored in network routers (i.e. network state) while the stack depth affects the needed IP packet header size, i.e. a very similar tradeoff to ours but in wired networks.

III. Problem Formulation

In this section we formulate the trajectory approximation problem as a combinatorial optimization problem. We note that there are a number of curves that can be used to approximate a trajectory, for example straight lines, bezier curves etc. We assume that different portions of the curve can be represented (approximated) using different curves. We first present some examples to motivate the trajectory approximation problem considered later in this section.

A. Examples

We note that there are many different ways of representing the trajectory itself, and do not delve into the details here. We are concerned with the efficient approximation of the trajectory under consideration using different types of curves available. For example, the trajectory can be approximated using (a possibly large number) of small line segments, resulting in an accurate

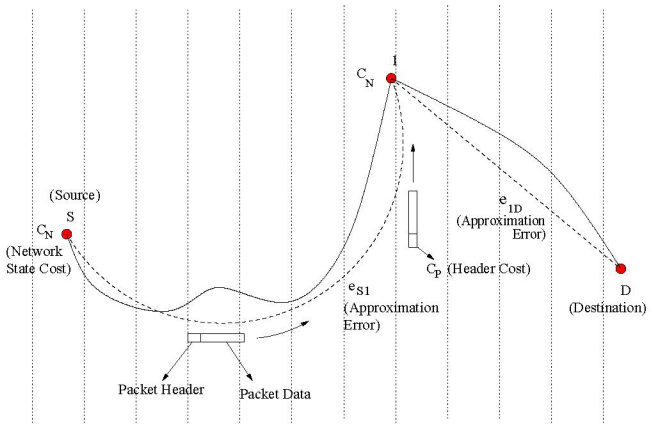

Figure 1: Example 1: Shows a trajectory split into two pieces, each piece is approximated using different curves.

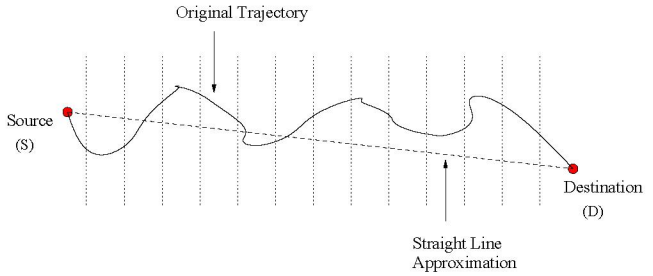

Figure 2: Example 2: Shows a trajectory being approximated reasonably accurately, with a simple straight line.

piecewise linear representation of the curve. We are interested in a minimum cost accurate representation of this trajectory.

Figures 1 and 2 show examples which explain the details of the mapping process described earlier. The vertical dotted lines in the figures represent the discretization interval (in space). Each intersection of these vertical lines (along with the source and destination points) represent a vertex in the graph representation of the trajectory. In Figure 2 the trajectory is approximated using a single straight line between source and destination points. This simple approximation of the trajectory in Figure 2 can be tolerated so long as the error due to the approximation does not exceed the prescribed bound. We note that the straight line approximation is cheap in terms of network state. In Figure 1 the trajectory is split into two portions (at a vertex in the corresponding graph representation). One portion is approximated using a curve and the other by a straight line. The example in Figure 1 shows a possible solution to the approximation problem when multiple types of curves can be used in the approximation. As compared to the example in Figure 2, one vertex in the graph representation is used to split the trajectory into two portions and each portion is approximated by using a curve from the available selection of curves. This split is to ensure that the error upper bound is not violated. While the example shows the first portion of the trajectory represented using a curve, we note that it's representation using a straight line will also result in a more accurate representation as compared

to single straight line between the source and destination.

B. System Model

When a trajectory is approximated by a series of curves, in the worst case each portion of the trajectory is associated with some cost in terms of packet header length, and network state. Note that depending on the packet header length, it may be possible to encode information about multiple portions of the trajectory. This may be the case for multiple *easily* representable portions of the trajectory. We make the following modeling assumptions:

- There are k choices for representing a given portion of the curve. These are denoted as $r_1, r_2, ..., r_k$. Here r_i is a straight line or a bezier curve etc.
- The space in which the trajectory exists is discretized, and there are a maximum of m points (excluding source and destination), or $m+1$ portions into which the trajectory can be split up.
- Since each portion of the curve is represented using a single type of representation, we construct a matrix Q of dimensions (m, k), where Q_{ij} is a binary variable, which denotes which representation is used for a particular portion of the curve. This implies that only a single entry in a row will equal 1.
- If the algorithm selects a particular representation r_i for some portion of the curve, then it makes use of a subroutine to compute the error associated with the representation, at unit cost.
- There is a header overhead cost C_p and a network state cost C_N for each portion of the curve, depending on the curve used to represent the portion of the trajectory being considered.
- Network state is maintained at the points along the trajectory which divide it into portions which are represented using different curves. These points (sensor nodes) maintain information about the approximation to the next successive portion of the trajectory to be represented.

We can now formulate the following binary program:

$$\min \sum_{i=1}^{m}\sum_{j=1}^{k} C_p(j)Q_{ij} + C_N(j)Q_{ij} \quad (1)$$

Subject to: (2)

$$\sum_{i=1}^{m}\sum_{j=1}^{k} e(Q_{ij}) \leq E \quad (3)$$

$$\sum_{j=1}^{k} Q_{ij} \leq 1 \;\; \forall\, i = 1 \cdots m \quad (4)$$

$$Q_{ij} \in B \quad (5)$$

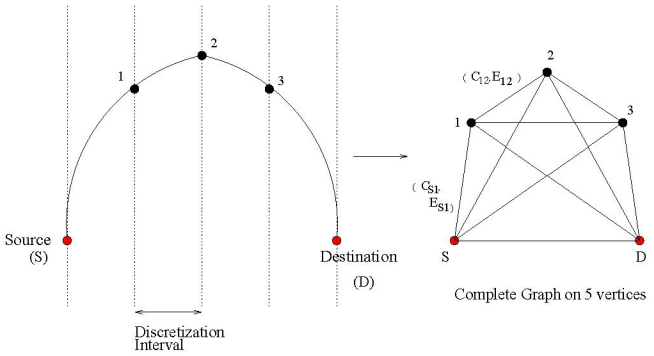

Figure 3: Example of Graph Representation of Discretized Trajectory

C. Formulation Details

In the formulation $(1 \cdots 5)$, the constraints (3) denote the error associated withe the representation of each portion of the curve. Note that of the m points available, we do not necessarily select all of them, this is captured in constraints (4). In constraints (3) the sum of the errors must not exceed an application defined error E which we assume is an input to the problem. Constraint (5) states that Q_{ij} is a binary variable. Note that the formulation can modeled using a graph on $m+2$ vertices (including source and destination nodes), with edges between all nodes (complete graph on $m+2$ vertices) implying that any of these edges can be chosen in an approximation of the trajectory, subject to the error constraints. Note that this implies that the type of approximation of the curve between 2 given vertices (the type of curve used) is fixed. For example we can assume that only straight line approximations of the trajectory are allowed.

To make the graph formulation more general, we can allow for multiple edges between two vertices, each edge corresponding to one type of approximation for the corresponding portion of the curve.

The edges are associated with an error measure as well as a cost due to C_p and C_N. While the formulation in $(1 \cdots 5)$ is *node* based, the edge formulation is implicit. For example a solution which selects some $l \leq k$ equal to 1 for some node i, and all other entries in the matrix as 0, implies that the approximation are the edges (s, i) and (i, t) where s and t are source and destination nodes respectively. We note that this problem is identical to SHORTEST-WEIGHT CONSTRAINED PATH, which is a well known NP-Complete problem, for example refer [12]. This is represented diagrammatically in the Figure 3. In Figure 3, each edge is associated with a cost as well as a representation error. This is represented on the edges (S,1) as (C_{S1}, E_{S1}) and (1,2) as (C_{12}, E_{12}) , though all edges are associated with such numbers.

D. A Simplified Version of the Problem

We note that in the absence of the error constraints, the problem becomes simple to solve. If header size induced by the complexity of representation of a portion of the curve is measured in bits, then it is reasonable to assume that the network state

is also a function of this number of bits used. Using the same graph representation as earlier, we associate each edge of the graph with a cost which is the sum of the header size cost and networks state cost incurred at the *source* node of the link (the trajectory, or its approximation is directed from source to sink). Note that we can construct a multigraph with multiple edges resulting from multiple possible representations of a given portion of the curve. Finding the minimum cost incurred by the representation is then equivalent to finding the shortest path in this graph which can be done using Dijkstra's algorithm.

E. Form of the Objective Function

The terms C_p and C_N in the objective represent the cost incurred for selecting an approximation for a given portion of the curve. These can be represented using bits. For example, how many bits of data need to be stored in the packet header and/or in an intermediate node. An actual function for these terms may be constructed based on battery power expended for transmission of bits in a packet header (we note that communication is much more expensive than computation in networks of limited capability devices), and buffer space consumed when storing network state, since memory is again an expensive resource when low end sensor nodes are considered.

F. Error Measures for Trajectory Approximation

In order to determine quality of trajectory approximation, we need to have a quantification of the approximation error. Given a particular splitting of the complete trajectory into pieces, we make this quality quantification by means of an aggregate error in representing the complete trajectory, which is the sum of the error in representation of each piece

Curve fitting is a well-studied problem. Fitting of parametric (e.g. Bezier or B-spline) curves relates to our optimization framework, since we will use an trajectory approximation error constraint. To have such an error constraint, we argue that two different error measures can be used: (i) deviation area per unit distance traveled along the trajectory, and (ii) normalized path length. One crucial point here is that we need error measures that can be expressed in a generic manner, i.e. independent of the shape or length of the particular trajectory at hand. For example, instead of "meters" we need a generic "percentage" value, such as 30% of error.

A generic way of expressing the trajectory approximation error requires capability of defining and calculating a *maximum* error amount. We are assuming that the representation methods (e.g. a line, single-point Bezier curve or cubic Bezier curve) are fitted to the trajectory in the best way , i.e. the control point(s) of the Bezier curve is/are adjusted so that the approximation error is minimized. So, the maximum error happens when the representation method is a simple line.

Let $L(s,d)$ be the straight line between the source s and the destination d. Given a trajectory, $T(s,d)$, and its approximation, $T'(s,d)$, we define two error measures for quantifying a percentage error bound E as follows:

1) Deviation Area

Deviation area, D, is the area between the actual trajectory and its approximation.

Let $e_{DA}(T(s,d),T'(s,d))$ be the deviation area between the trajectory $T(s,d)$ and its approximation $T'(s,d)$. The maximum deviation area happens when the approximation is a straight line, i.e.

$$e_{DA}^{max}(T(s,d)) = e_{DA}(T(s,d),L(s,d))$$

This means that, in order to fulfill the the percentage error bound E, the $e_{DA}(T(s,d),T'(s,d))$ value must satisfy the following inequality:

$$e_{DA}(T(s,d),T'(s,d)) < \frac{E}{100}\ e_{DA}^{max}(T(s,d))$$

2) Normalized Length

Let $||T(s,d)||$ be the length (in meters) of the trajectory $T(s,d)$. We measure the normalized length error $e_{NL}(T(s,d),T'(s,d))$ by:

$$e_{NL}(T(s,d),T'(s,d)) = \left|1 - \frac{||T'(s,d)||}{||T(s,d)||}\right|$$

The maximum normalized length error for $T(s,d)$ is:

$$e_{NL}^{max}(T(s,d)) = \left|1 - \frac{||L(s,d)||}{||T(s,d)||}\right|$$

This means that, in order to fulfill the the percentage error bound E, the $e_{NL}(T(s,d),T'(s,d))$ value must satisfy the following inequality:

$$e_{NL}(T(s,d),T'(s,d)) < \frac{E}{100}\ e_{NL}^{max}(T(s,d))$$

Based on this inequality, once a percentage error bound E is given, we can trace it into the maximum possible normalized length error in the trajectory approximation.

IV. Heuristics to solve the Constrained Shortest Path Problem

A. Trajectory Partitioning Heuristic

This heuristic is based on exhaustive search of the paths in the graph representation. The curve is first split into 2 parts. The node defining the split point is one which has it's x coordinate half the difference in the x coordinate of the source and destination point respectively. Dividing the curve into multiple parts can be similarly done. Now, starting from the source, the first portion of the curve is approximated as closely as possible, that is, with minimum error possible. This is continued for all portions of the curve till the last portion of the curve is reached. Here, instead of using the minimum error curve approximation, the worst approximation which does not violate the overall error limit can be used. Consider the case when the trajectory is split into 2 parts. If even the best approximation for both portions violates the error constraint, then this is not a feasible solution. We note that this heuristic will lead to a

good initial approximation of the trajectory, but the last portion of the trajectory will not be as accurately represented. We note that the full details of this heuristic are not presented here due to space constraints.

B. Equal Error Heuristic

This heuristic divides the trajectory into a number of equal parts using the same strategy as the previous heuristic, but takes an equal performance hit in terms of deviation from the best approximation of a portion of the curve. For example, when the curve has four portions, each portion is approximated as close to 25% error as possible. We note that the herusitic can be implemented in a manner similar to the heuristic presented in the earlier subsection.

V. Summary and Future Work

A. Summary

In this paper we have presented a problem related to the implementation of trajectory based routing, and under certain abstractions of the system, have formulated a combinatorial optimization problem which we show is difficult to solve to optimality. We therefore present two heuristics to solve the problem, which work well under assumptions on the input to the problem.

B. Future Work

The models and formulations presented in this work suggest a number of problems which require more detailed investigation. For example, a similar investigation to the one presented in Section B. can be done for the case where the system operates with a fixed header size. While the model in Section B. is more flexible in that variable size headers are allowed, and larger size headers will be penalized in terms of cost, it is likely that real world systems will operate with fixed header size. This leads to two effects, first that a fixed header size implies that the cost of encoding a spatially simple curve is the same as that of encoding a more complicated curve, and second that multiple simple trajectories can possible be encoded in a fixed header size at the same cost, with the added benefit of reduced network cost. We note that the techniques presented in this paper show that in general, the centralized version of the problem is also hard to solve. However, distributed heuristics are of interest since the problems are typically in a sensor network setting.

References

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

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

[3] 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.

[4] N. Bulusu, J. Heidemann, and 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.

[5] D. Niculescu and B. Nath, "Trajectory based forwarding and its applications," in *Proceedings of ACM MOBICOM*, 2003.

[6] M. Yuksel, R. Pradhan, and S. Kalyanaraman, "An implementation framework for trajectory-based routing in ad hoc networks," *Ad Hoc Networks*, vol. 4, no. 1, pp. 125–137, January 2006.

[7] A. Kolesnikov, P. Franti, and X. Wu, "Multiresolution polygonal approximation of digital curves," in *Proceedings of IEEE International Conference on Pattern Recognition (ICPR)*, 2004.

[8] P. L. Rosin, "Techniques for assessing polygonal approximation of curves," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 6, pp. 659–666, 1997.

[9] B.B. Chaudhuri and S. Dutta, "Interactive curve drawing by segmented bezier approximation with a control parameter," *PRL*, vol. 4, pp. 171–176, 1986.

[10] V. S. Mirrokni, M. Thottan, H. Uzunalioglu, and S. Paul, "A simple polynomial time framework for reduced-path decomposition in multi-path routing," in *Proceedings of IEEE INFOCOM*, 2004.

[11] A. Gupta, A. Kumar, and R. Rastogi, "Exploring the trafe-off between label size and stack depth in mpls routing," in *Proceedings of IEEE INFOCOM*, 2003.

[12] Editors Pierluigi Crescenzi, Viggo Kann, "A compendium of np optimization problems," *http://www.nada.kth.se/ viggo/problemlist.*