CAR: Controlled Adjustment of Routes and Sensor Networks Lifetime

Goce Trajcevski* Oliviu Ghica Peter Scheuermann†
Department of EECS
Northwestern University
Evanston, Il, USA
{goce,oliver,peters}@eecs.northwestern.edu

Abstract

This work addresses the problem of extending the lifetime of a wireless sensor network, when a bounded delay on receiving the packets is acceptable for a given sink node. The temporal threshold for the Quality of Data (QoD) tolerance is specified with respect to the time that it takes for the packets to travel from the source to the sink along an optimal route. For these settings, we propose a methodology that enables controlling the spatio-temporal load balancing among the sensors around the optimal route. Given the QoD threshold, we use it to derive a set of parameterized Bezier curves which serve as alternate routes between the (source,sink) pair, relieving the nodes along the optimal route, while ensuring a bound on the delay of packets delivery. We provide experimental results which demonstrate that our proposed methodology can prolong the sensor networks' lifetime for various definitions of the concept of a "lifetime" in the existing literature.

1. Introduction and Motivation

Wireless sensor networks have recently become a very popular paradigm that has a wide range of potential applications [22]. The ability to deploy a large amount of relatively cheap sensor nodes that are capable of measuring particular physical values, perform calculations and, most importantly, organize themselves in a network, has spurred a tremendous amount of scientific and industrial research. One of the tightest constraints of the sensor nodes is their limited on-board energy and many researchers have addressed the problem of maximizing the lifetime of the sensor networks from various perspectives [5].

One body of works has specifically focused on development of efficient routing protocols (data-centric; hierarchical/topological; location-based) [1] and, in a sense, the work presented here belongs to this category. The motivation for our work can be traced back over 2000 years and is expressed by the fable typically attributed to Aesop – "The Selfish Horse": *A farmer set off on journey with many baskets, taking both his horse and his donkey to carry the load. But after many miles, the little donkey could no longer carry all his burden. He begged the horse to help him, but the horse was quite comfortable as he was and refused. Soon the exhausted donkey could walk no further and he collapsed in the road. The farmer, who had no intention of stopping, put all the donkey's baskets on the horse's back and continued on his way. "I wish I had helped the donkey earlier," wailed the horse. "Now I have to carry the whole burden alone."* The obvious moral of the story is that it pays off to do a little help on behalf of others.

Typically, when a given node (sink) requests a data sensed by another node (source), some sort of an *optimal* route is established between them [11, 16, 19, 20]. In our setting, we consider *(sink,source)* pairs for which there is a long-running/continuous query that requires constant transmission of the measured/sensed data. The nodes along the optimal route are the donkeys, and we call them γ-nodes (from the Greek $\gamma\alpha\iota\delta\upsilon\rho\iota$ = donkey). The sensors around them are the horses, called α-nodes ($\alpha\lambda o\gamma o$ = horse). Hence, the problem that we address in this paper is how to spatio-temporally distribute the load among the α-nodes and the γ-nodes, so that the overall lifetime of the network is prolonged as much as possible. But then the question becomes *what is* a *lifetime* of a given network? There are six different definitions surveyed in [5] like, for example: - time until the first node dies; - time until a certain fraction of the nodes die; etc. We observe that regardless of the definition of the lifetime, we cannot afford to send the packets along purely random routes or, for that matter, have too long α-nodes routes, because that may impose an unacceptable delay for the packets reception at the sink.

This is precisely the main contribution of our work. We propose a methodology that enables a development of a set

*Research supported by the Northrop Grumman Corp. grant PO 8200082518

†Research supported by the NSF grant IIS-0325144/03

of routes consisting of α-nodes that we can use to alternate among and partly relieve the optimal route of the γ-nodes, in a controlled manner. Similarly to [16], we combine the sink-based decision about the properties of the route with the local decisions of the nodes based on their relationship with the route, however, in this work we generate a *set* of routes. Furthermore, as a QoD constraint, one can impose a bounded delay on the packets reception and the value of this bound can be used as a parameter when deciding upon the properties of the routes. Thus, in a sense, we are taking a step towards explicitly tying some QoD-related criteria [12, 3] with the lifetime of the network as a whole. We developed a simulator that is built on top of the JiST SWAN [10] and conducted extensive experimental evaluations of our ideas. Our experiments demonstrate that for different definitions of the network lifetime, our approach can extend it and the trade-off with respect to the extra energy-expenditure is acceptable.

The rest of the paper is structured as follows. In Section 2 we provide the necessary background. Section 3 presents the main analytical aspects of our approach and presents the algorithms executed by the nodes for the controlled management of the routes. In Section 4 we present our experimental observations. Section 5 positions our work with respect to the relevant literature and Section 6 concludes the paper and proposes directions for future work.

2. Preliminaries

We assume that we have a large number of sensor nodes which are deployed randomly in the geographic area of interest and independently of each other's location, i.e., they follow a Poisson distribution [17, 22]. Each node is aware of the values for the *(x,y)* coordinates of its location, which is obtained either by a GPS or by initially communicating with a set of beacon nodes (c.f. [22, 1]). Furthermore, each node is also aware of the location of its "one hop" *neighbors*, i.e., the set of nodes within its communication range.

In order to understand the main technical aspects of our approach, we need to explain the concept of a Bezier curve. Without loss of generality, we assume that we are dealing with the 2D Euclidian space.

Given $n+1$ points: P_0, P_1, ..., P_n called *control points* and a parameter[1] $u \in [0,1]$, the *Bezier curve* is defined as the set of all the points:

$$C_b(u) = \sum_{i=0}^{i=n} B_{i,n}(u) \cdot P_i \qquad u \in [0,1]$$

where $B_{i,n}(u)$ are the *Bernstein polynomials*, defined as $B_{i,n}(u) = \binom{n}{i} u^i (1-u)^{n-i}$ for $i \in \{0,1,\ldots,n\}$ and $B_{i,n}(u) = 0$ otherwise [6]. Thus, a curve with $n+1$ control points is represented as a polynomial in u of n^{th} degree. A more general class of curves called *splines* is also used in computer graphics and computer-aided geometric modeling of curves and surfaces [6]. However, in this work we restrict ourselves to Bezier curves only, for the following reasons: they are the simplest (in terms of algebraic representation) splines; they have been extensively studied and tools exist which can generate the respective Bezier approximation for curves drawn by the users on a screen. Thus, as a first step, they provide a good foundation to explore the validity of our ideas.

Bezier curves have a number of important properties [6], such as:
1.) *convex hull:* – all the points on the curve are contained in the convex hull of the control points;
2. *pseudo-local control:* – moving one control point has predictable effects on the changes of the curve's shape and those effects are strongest around that point.
3.) *Endpoint interpolation* – the curve always passes through the end-control points P_0 and P_n and, moreover, the segments P_0P_1 and $P_{n-1}P_n$ are tangents to the curve.
4.) *Affine invariance* – in order to apply a particular affine transformation to the points on the curve (i.e., for all the values of u), it suffices to apply those transformation to the control points and apply any standard algorithm, e.g., De Casteljau [6] for generating the (points of the) Bezier curve using the new set of control points for any given u. In particular, if the m^{th} control point P_m is translated by a vector $\vec{v}$, then for each value of the parameter u, the corresponding point will be translated by a collinear vector, properly scaled at the m^{th} Bernstein polynomial:

$$C_{new}(u) = C(u) + B_{m,n}(u) \cdot \vec{v}$$

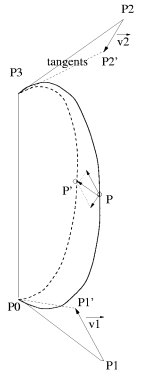

Figure 1. Bezier curve and its properties

The properties that are more interesting for us are 3.) and 4.), and they are illustrated in Figure 1. It shows the curves of order 3, which have 4 control points. The wider curve is

[1] If $u \in [a,b] \neq [0,1]$, linear scaling can be applied to the interval of interest, and a new variable $u' = \frac{u-a}{b-a}$ can be used.

the "original" and one can see the tangents at the end-points. The inner (dashed-lines) curve is the one that is obtained when the control points of the original curve are translated to new positions. Observe that, when the control points P_1 and P_2 are translated to their new locations P_1' and P_2' by vectors $\vec{v_1}$ and $\vec{v_2}$, the point P on the curve is moved to its new location P' on the new curve. For a given value of u, the effect of the deformation is actually a translation by a vector that is a linear combination $B_{1,3}(u)\vec{v_1} + B_{2,3}(u)\vec{v_2}$.

Another result that we will use in Section 3 is presented in [2]. A discrete approximation of the Bezier curve, can be obtained as follows: let $j \in \{1, 2, \ldots, k\}$ and let $u_j = j\delta$, where $k\delta = 1$, i.e., we discretize the interval $[0, 1]$ into k equal sub-intervals of size δ. If we use the values of u_j's only and approximate the rest of the curve (the other values of u) by straight line-segment between two consecutive discrete values (e.g., u_j and u_{j+1}), then the approximation can be specified as the set of points:

$$\hat{C}_b(u) = \sum_{i=0}^{i=n} \hat{B}_{i,n}(u) \cdot P_i \qquad u \in [0, 1] \text{ where:}$$

$$\hat{B}_{i,n}(u) = \begin{cases} B_{i,n}(u) & \text{for } u = u_j, \text{ and} \\ B_{i,n}(u_j) + (u - u_j)\frac{B_{i,n}(u_{j+1}) - B_{i,n}(u_j)}{u_{j+1} - u_j} & \\ & \text{for } u \in (u_j, u_{j+1}) \end{cases}$$

Lemma 1 in [2] ensures that in this case the maximum distance from the point on the curve and its corresponding (with respect to the value of u) point on the polyline, is bounded by $\frac{1}{\delta^2} n(n-1) \lfloor \frac{n+1}{2} \rfloor diam(P)$, where $diam(P)$ is the diameter of the set of control points; n is the degree of the curve ($n + 1$ control points) and δ is the step of the discretization of the interval $[0, 1]$.

3. CAR and Load-balancing Among Multiple Routes

Typically, when a particular *sink* node requests continuous readings of the values obtained by a given *source* node, some protocol is used to establish an *optimal* route between the source and the sink, possibly combining the knowledge of the geography and/or topology with the one regarding the energy in the individual nodes [11, 15, 16, 19].

When the sink decides that it needs the readings from a particular geographic location, since it knows its own location, it can calculate the approximate length of the optimal route[2]. Subsequently, by using some typical values for the processing-delay and packet-drop rate at an individual node, the sink can estimate the delay between the time at which a particular packet has been sent by the source and the time it receives that packet. The crucial observation is that, since the temporal delay is directly correlated with the length of the route used, if the sink can tolerate a delay greater than the one along the optimal route, then the length of the curve corresponding to the longest acceptable delay can be determined.

Let L_o denote the approximated length of the optimal route and let L_{max} denote the approximated length of the longest route whose usage induces an acceptable delay for the packets. Then, any route whose length L_l satisfies $L_o \leq L_l \leq L_{max}$ can be used as an alternate route for the packets from the source to the sink. However, *randomly* generating the alternate routes at "run-time" may be too costly in terms of communication among the nodes and further unexpected delays and, moreover, does not provide any means to control the choice of routes. Furthermore, the sink may also want to specify (some properties of) the *shapes* of the routes. For example, although motivated by a different problem, [21] pointed out that the nodes around the sink are more frequently used and have higher probability of interference. Thus, the sink node may choose to "wrap" some more nodes in its immediate neighborhood.

Our methodology enables the specification of the desired shape of the routes and their construction which can be done along with the construction of the optimal route for a given (*sink,source*) pair and without too much of an communicated-information and computational overhead. Each node locally determines which route it belongs to which, in turn, enables a local decision-making for selecting a transmission along a particular route.

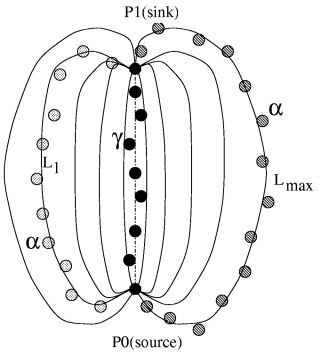

Figure 2. Alternate Routes

Figure 2 depicts an optimal route (dark circles for the γ nodes) and a set of alternative routes consisting of α nodes (gray circles) around it. Now we proceed with explaining how to achieve the desiderata discussed in the previous paragraphs.

[2]Throughout this work, an optimal route is the one that corresponds to the shortest path between the source and the sink

3.1 Constructing the Routes

Similarly to Section 2 and, again, without a loss of generality, in the sequel we will assume that $L_o = 1$, and every other length/distance is properly scaled. There are three main aspects of construction of the set of routes:

I - Boundary Route Construction: First, we fix the end-control points of any Bezier curves to be *sink = (0,1)*= P_n and *source = (0,0)* = P_0. Subsequently, given the desired shape, the sink selects the rest of the control points and the discretization step δ of the interval [0,1] ($k\delta = 1$). Then, the sink can approximate the length of the boundary route as: $L_{max} = \sum_{j=1}^{j=k} \overline{\hat{C}_b(u_j)\hat{C}_b(u_{j-1})} = \sum_{j=1}^{j=k} \sqrt{(\sum_{i=0}^{i=n} B_{i,n}(u_j)P_{ix} - \sum_{i=0}^{i=n} B_{i,n}(u_{j-1})P_{ix})^2 + + \sum_{i=0}^{i=n}(B_{i,n}(u_j)P_{iy} - \sum_{i=0}^{i=n} B_{i,n}(u_{j-1})P_{iy})^2}$ where P_{ix} and P_{iy} denote the x and y coordinates of the control point P_i.

Obviously, in reality, one would need to check whether L_{max} introduces a delay which is acceptable for the sink and may need a few iteration with adjusted values for the control points.

II - Alternate Routes: Once an acceptable boundary route has been calculated, we use the affine invariance property to generate a family of Bezier curves, each corresponding to an alternate route. To do so, we view the boundary route and the optimal route as two extremes. The process of "degenerating" the boundary route to the optimal one can be achieved by projecting the set of control points on locations along the optimal route. To select those locations, in this work we adopted an approach that we call *proportional projection*, which can be explained as follows. Given the initial locations of the control points, their terminal locations on the $\overline{P_0, P_n} = \overline{(0,0),(0,1)}$ line connecting the sink and the source are the points which:

1.) preserve the original order among the control points;

2.) preserve the ratio of the distances among the line segments connecting two consecutive control points.

Observe that other rules may be applied to determine the final locations of the control points, e.g., they can be uniformly distributed along $\overline{P_0P_1}$.

Now, we only need to select the number of discretization steps along the vector connecting the initial position of a given control point P_i, with its final position P_i^f. Throughout this work, we assume that the total number of steps is the same for every control point P_i, ($i \in \{1, 2, \ldots (n-1)\}$), and we denote it by m (see Figure 3 for illustration).

The process of establishing the routes can now be explained as follows. After determining the acceptable set of control points and the granularity-level m for the number of the alternate routes, the sink broadcasts the "regular" request for generating an optimal path between itself and the source-location, except it adds the following parameters to the request:

- the initial and the final positions of each control point;
- k - the number of steps used in discretizing the parameter u of the Bezier curves; and
- m - the number of steps used in discretizing the locations of the control points;

III – Node-to-Route Mapping: After receiving a request for establishing a route, each node needs to determine its closest route, as well as its (approximate) position along it. Both values are, in a sense, intertwined and, since there are m possible routes and each of them is a piecewise approximation of the Bezier curve with k segments, it seems that the complexity of the computations executed in a particular node is $O(km)$.

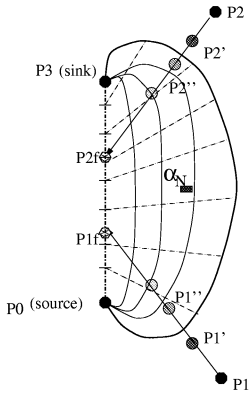

Figure 3. Scaling and closest curve determination

However, a closer observation reveals that, for a given node, this amounts to determining separately:

1.) Which value of u is the one that generates the closest point on (one of) the Bezier curves from the family of alternate routes.

2.) Subsequently, which one is that actual closest Bezier curve.

Figure 3 depicts a set of routes and a node α_N for which the node-to-route mapping needs to be done. α_N uses the affine invariance property to determine the value of $j \in \{1, \ldots, k\}$ for which its corresponding u value on the closest curve is guaranteed to be $u_{j-1} \leq u_{\alpha_N} \leq u_j$. To describe how α_N does this, recall that it has received the parameters specifying the initial and final positions of the control points, as well as the values k and m, transmitted along with the request for forming a route. Thus, α_N can reverse the directions of the vectors describing the movement of each control point P_i. By doing so, it calculates the rays originating in each discrete u_j and describing the possible locations of the vertices for the polyline approximations of the Bezier curve. The closest point to α_N on any $\hat{C}$ is either at $\hat{C}_{u_j}$ or along the line segment $\overline{\hat{C}_{u_{j-1}}\hat{C}_{u_j}}$.

However, determining this can be done in $O \log k$ by a binary search. Once it has determined the "strip" to which it belongs (i.e., the values of j), α_N needs to determine which is the particular curve of the family representing the possible routes which is closest to it. However, this reduces to determining the value of $l \in \{1, 2, \ldots, m\}$, such that the line segment $\overline{\hat{C}^l_{u_{j-1}} \hat{C}^l_{u_j}}$ (or one of its end-points) is closest to α_N. This, again, can be executed in $O(\log m)$ using binary search. Hence, the total complexity of determining the node-to-route mapping executed locally by a given node is $O(k + \log k + m + \log m) = O(k + m + \log km)$.

3.2 Alternating Among the Routes

Upon establishing the routes, the source node has all the information about them, in terms of the corresponding control points and the parameters k and m. Once a sensed value needs to be transmitted, the source node decides which alternate route is to be used for the particular packet and, along with the rest of the information, it sends the value of $l \in \{1, \ldots, m\}$ which is the "index" specifying the alternate route to be used for that packet.

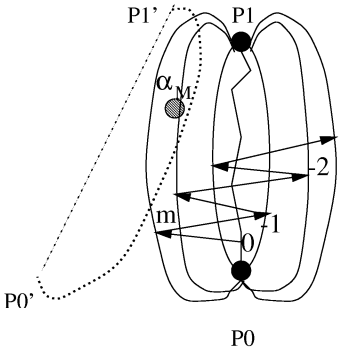

Figure 4. Alternating Among the Routes

Assuming that the routes are symmetric around the shortest path between the source and the sink, and adopting the notation that the (left) counter-clockwise direction is a positive one, the set of alternating routes can be enumerated as $S = [m, (m-1), \ldots, 1, 0, -1, \ldots, -(m-1), -m]$, where "0" corresponds to the optimal route. The policy of alternating among the routes now amounts to selecting the permutations of S to be used in transmitting the packets[3]. One may be tempted to use the simplest permutation for periodic alternation along the sequence which, in one period, selects the routes based on the order in the original sequence S. We call this a *sweep* based alternating among the routes.

However, as observed by the other researchers (e.g., [21]), and as we experimentally demonstrate in Section 4, there may be packets-loss due to interference among the nodes along spatially close routes. In our settings, when the sampling frequency is high, and a packet is transmitted for every sampled value, spatially close nodes will be employed for different routes in small time-intervals. When the *sweep* approach is use to select among the candidate routes, this will result in an interference which is especially observable in the proximity of the sink node. Motivated by this, we propose the permutation for alternating among the routes which, during one period, corresponds to the following sequence: $m_d = [0, m, -1, (m-1), -2, (m-2), -3, \ldots, 2, -(m-1), 1, m]$. We call this the *constant mid-distance* permutation for alternating the routes. It is illustrated in Figure 4 by the arrowed polyline indicating the sequence of routes' selection among the source (P_0) and the sink (P_1).

One may, rightfully so, observe that a particular γ and/or α nodes may be part of more than one family of routes because there may be more than one continuous queries pending in the network, between different *(sink,source)* pairs. In the scenario depicted in Figure 4, this is illustrated by the dotted line between P'_0 and P'_1 and the node α_M, which now needs to serve two alternate routes for two different continuous queries. However, this, may introduce an additional delay along both of the routes that are using it. In the current stage of our work, we handle this in a simplistic manner:

1.) If the node is a γ-node, it is not allowed to relieve itself from the current route.

2.) If the node is an α-node, upon estimating that it introduces a significant delay on the route that it belongs to, it signals a notification that it can no longer serve along all the routes. Once the source of a particular route receives such notification, it discards that route from the set of the available ones. When the node is of an α type for more than one query, it selects to remain part of the alternate routes for the query whose optimal route is closest to it. In the scenario depicted in Figure 4, the node α_M will continue to serve the query between P'_0 and P'_1.

4 Experimental Results

Now we present the experimental evaluation of our proposed approach. Our experiments were conducted using an Intel Pentium IV 3.2 GHz processor with 2GB of DDR2 RAM. We built our own SNSim simulator[4] on top of the java based JiST-SWANS [10] simulator as a core. We used 600 nodes deployed in a 2 x 2 km^2 area, which used 802.11 protocol at the MAC layer, and the heartbeat node discovery protocol in order to determine the neighbors. As part of our SNSim, we encoded the "electrical" parameters of the

[3]Observe that the source may "randomly" choose a given route, and we present our experimental observations for this case in Section 4.

[4]The java source code is publicly available at http://www.eecs.northwestern.edu/peters/sensors

nodes (e.g., the energy consumption model) so that their behavior mimics the MICA Motes (www.xbow.com). Our limit on the time-delay along the boundary route with respect to the optimal route for a given query was 50%.

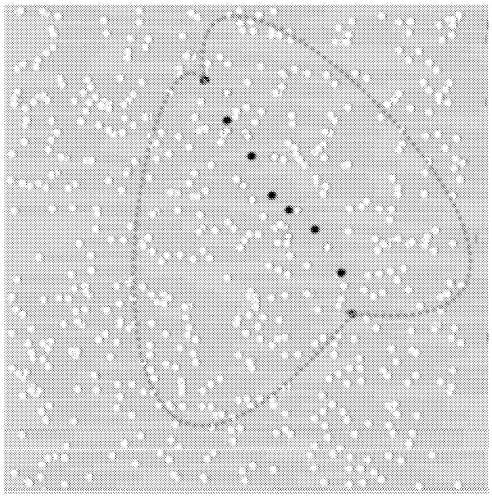

Figure 5. Snapshot of the Network

Figure 5 gives a zoomed-snapshot of an instance of the network, as displayed by the SNSim. The white circles represent the sensor nodes, the black circles indicate the trace of the optimal route, and we depict two Bezier curves of possible boundary routes with dashed lines.

The duration of monitoring a continuous query for the purpose of evaluating the energy loss was 20,000 packets between the source and the sink, and we varied the sampling (equivalently, transmission) rate between 0.1 sec. and 20 sec. Thus, in effect, we simulated some long-running queries in excess of 100 hours, for small sampling frequencies. One of the reasons we varied the sampling frequencies is because we wanted to observe the effects of packets-loss in the nodes due to interference among the alternate routes. We used the following approaches for alternating among the routes between a given (*source,sink*) pair (c.f. Section 3.2): 1.) Shortest-path route (no alternating); 2.) Bezier Curves-based: random choice of routes, simple sweep, and constant mid-distance alternating.

Figure 6 presents the energy left in the entire network when a continuous query between a given source and sink is run for up to 30 hours, using a particular route-selection policy. The graphs are obtained by averaging the values for 20 random queries, each with 5 sec. sampling frequency. Observe that we are not dealing with the lifetime here, and the main purpose of this result is to demonstrate that the extra energy-expenditure of the alternating routing is close to the one used by a single optimal route. Even when a query is run for a long time, the extra-energy cost over the whole network is only 5% more than the one incurred by

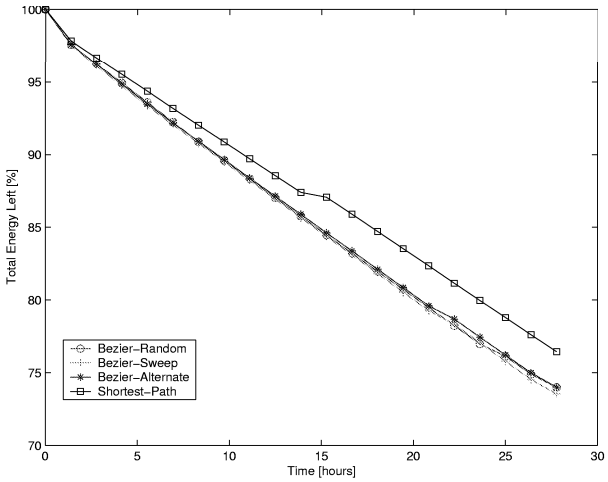

Figure 6. Energy Consumption

using the shortest-path route.

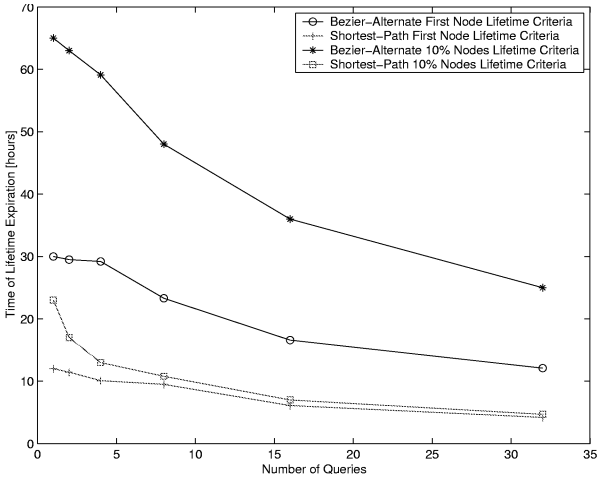

Figure 7. Lifetime Extending via Alternating Routes

To get the first quantitative idea about the benefits of alternating the routes for the network lifetime, observe Figure 7. Each curve depicts the duration of the lifetime of the network when multiple continuous requests, whose number is represented on the x-axis, are simultaneously present. The results were averaged over 10 simulations. We used two definition for the lifetime: 10% of the nodes depleted and the first node depleted, and the legend is given in the upper-right corner. Observe that for each definition of the lifetime, when alternating among the routes is used, the network lifetime is much larger than the one when only the shortest-path route is used between the respective sink and source. Combining with the results presented in Figure 6, we observe

that at a small extra-energy cost, the lifetime of the network can be significantly extended using alternating routes.

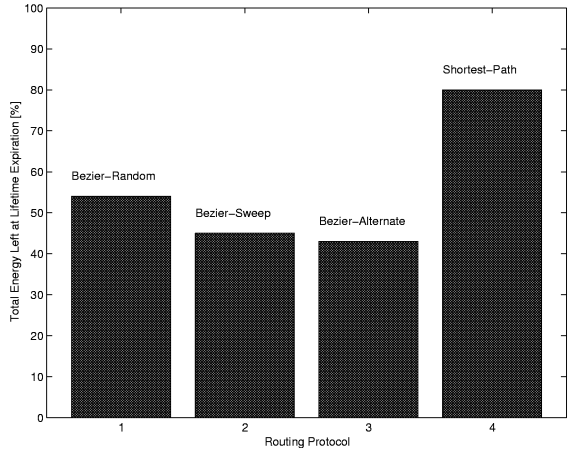

Figure 8. Lifetime vs. Energy Leftover

Figure 8 gives another quantitative description of the benefits of using the alternating routes. We assumed that the *lifetime* of the network is the time until 10% of the nodes die. The results in Figure 8 were obtained by averaging the results of 10 simulations, each running between 5 and 10 random queries simultaneously. The interpretation of the results is as follows: when using a single route (shortest-path) for the *(sink,source)* pairs, the network is declared "dead" (10% of the nodes died) even though 80% of its total energy for all the nodes is still left. In other words, the network dies and yet only 20% of its total energy is used. Clearly, when the constant mid-size alternating is used for selecting the routes, the overall usage of the network is maximal (equivalently, the energy left is minimal) at the time the 10% of the nodes have died. The reason that the sweep-based and the random alternating behave slightly worse is that each of them is more prone to packets-drop due to collisions. As one would intuitively expect, we obtained similar observations when the lifetime of the network is defined as the time until the first node dies, except the values for each routing method were almost proportionally higher (not shown).

5 Related Work

A very recent and concise survey that considers the complexities of several lifetime maximization problems in sensor networks settings, and for various energy consumption models, is presented in [5]. Another recent work which surveys routing protocols in wireless sensor networks is presented in [1]. Both of them provide extensive lists of recent results that are, in one way and level or another, related to our work. However, it is beyond the scope of this paper to do such exhaustive comparison and, in the sequel, we will position our results with respect to a selected subset of the relevant literature.

The thought of using multiple path for routing purposes in sensor networks is not new. Results (based on and) extending the Directed Diffusion approach [9] with considering multiple paths are presented in [7]. In addition to the disjoint paths which do not intersect, the work also considers *braided* paths which may intersect. Although some of the goals are similar to ours (energy efficiency), the work is more focused on resilience to failures and it proposes routings for which the decisions are made locally. In contrast, our work considers global properties of the routes, with local computation of the relevant values for a given node, based on the parameters describing the global properties. Moreover, since [7] is motivated/based on the directed diffusion, it is based on the query-driven data delivery model and may not be the most suitable choice for applications that require continuous data delivery, which is the case of our settings.

Algorithms for local construction of energy-efficient topology are presented in [19] where planar spanning structures with guaranteed stretch factors are presented. The structures utilize concepts such as Yao graphs and Gabriel graphs [13] to enable each node to make local decisions about selection of neighbors and communication power. As demonstrated further in [14], these structure guarantee packet delivery when localized routing methods are used. These works are, in a sense, orthogonal to ours but it would be interesting to investigate if the set of alternate routes can be built on top of a subset of the nodes specified by certain topological predicates.

The results in [21, 18] demonstrated that interference can be tackled and some controlled form of concurrent transmission can be achieved from a set of source nodes towards a given sink. The work also provided a formal algebra for specifying the concurrency/serialization of the routing. The problem that we addressed here is, in a sense, complementary to the ones addressed in [21, 18]. We consider a single sink-source pair, but we focused on generating a set of alternate routes that can be used in a controlled manner.

Our work is similar in spirit to [16, 15], which introduced the concept of a trajectory based forwarding (TBF). The work focuses on enabling a routing protocol where the nodes locally decide about forwarding of the packets, based on a pre-specified trajectory. In a sense, [16] discusses richer set of scenarios than we do (e.g., discovery of topology; broadcast; multicast) whereas we focus on a simpler setting of a long-running continuous query for a (*sink,source*) pair. Further generalization is provided in [15], which introduces the TBF+ concept, which uses polynomial curves for the trajectories (e.g., an ellipse). Using curve-fitting and the information about the energy of a par-

ticular node, TBF+ forces the trajectories to pass through points with higher energy reserves. We are using algebraic parameterized curves and we developed a formalism that enables constructing a family of such curves for the purpose of generating alternate routes. Furthermore, we provide a possibility to link the alternate routes with the QoD criteria of acceptable temporal delay while prolonging the lifetime [5] of the network. It would be interesting to combine our work with the ideas in [16] for the purpose of taking the current energy of a given node into consideration when constructing a particular alternate route.

6 Concluding Remarks

In this paper we used piece-wise linear approximation of Bezier curves to generate a set of alternate routes between a given sink/source pair. We presented an approach which enables efficient algorithms that can be executed locally by the individual sensor nodes for the generation of the routes. We demonstrated that one can control the alternation among the routes which yields an extension of the lifetime of the overall sensor network, even when different definitions of the concept of a lifetime are employed. Moreover, we also demonstrated that the CAR approach can be linked with at least one QoD criteria – the temporal delay of the packets.

Currently, we are investigating the problem of a more flexible adaptability of the family of curves representing the alternate routes (c.f. Section 3.2.), when multiple queries are posed within a small geographic region. There are several immediate extensions of our work. We are interested in extending our ideas when a given sink requests sensing from a geographic range. We are also investigating the problem of adjusting the selection of the nodes along a particular route (c.f. Section 3.1) and investigate its impact on the energy savings[5]. On a more general level, there are several interesting questions that could potentially use the approach that we presented here. Can some form of controlled alternating of routes be used for processing database queries in sensor network (e.g., in conjunction with [4])? Can our approach be incorporated in some algebra for specifying the desired properties of the routing (c.f. [21])? On a meta-level, an interesting problem is whether there can be other ways to link issues that belong to different "semantic dimensions", similarly to what we did for the temporal aspect of the QoD and the networks lifetime (c.f. [12])? We hope that some of these questions will be addressed in a near future.

References

[1] K. Akkaya and M. Younis. A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks*, 3(3), 2005.

[2] H. Alt, E. Welzl, and B. Wolfers. Piecewise linear approximation of bezier curves. In *Symp. Computational Geometry*, 1997.

[3] M. Bawa, A. Gionis, H. Garcia-Molina, and R. Motwani. The price of validity in dynamic networks. In *ACM SIGMOD*, 2004.

[4] C. Buragohain, D. Agrawal, and S. Suri. Power aware routing for sensor databases. In *INFOCOM*, 2005.

[5] Q. Dong. Maximizing system lifetime in wireless sensor networks. In *IPSN*, 2005.

[6] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, 1993.

[7] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-resilient, energy-efficient mutipath routing in wireless sensor networks. *Mobile Computing and Communications Review*, 5(4), 2001.

[8] M. Haenngi and D. Puccinalli. Routing in ad hoc networks: A case for long hops. *IEEE Communications Magazine*, Oct. 2005. Series on Ad Hoc and Sensor Networks.

[9] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed difusion: a scalable and robust communication pardigm for sensor networks. In *MOBICOM*, 2000.

[10] JiST – Java in Simulation Time / SWANS. http://jist.ece.cornell.edu, 2005.

[11] G. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless sensor networks. In *MOBICOM*, 2000.

[12] S. Li, S. H. Son, and J. A. Stankovic. Event detection services using data service middleware in distributed sensor netwroks. In *IPSN*, 2003.

[13] X.-Y. Li. Algorithmic, geometric and graphs issues in wireless networks. *Wireless Communications and Mobile Computing*, 3(2), 2003.

[14] X.-Y. Li, W.-Z. Song, and W. Wang. A unified energy-efficient topology for unicast and broadcast. In *MOBICOM*, 2005.

[15] M. V. Machado, O. Goussevskaia, R. A. F. Mini, C. G. Rezende, A. A. F. Loureiro, G. R. Mateus, and J. M. S. Mogueira. Data dissemination using energy map. In *WONS*, 2005.

[16] D. Niculesu and B. Nath. Trajectory based forwarding and its applications. In *MOBICOM*, 2003.

[17] P. Olofsson. *Probability, Statistics and Stochastic Processes*. Wiley-Interscience, 2005.

[18] D. Sharma, V. Zadorozhny, and P. Chrysanthis. Timely data delivery in sensor networks using whirlpool. In *DMSN Workshop*, 2005.

[19] W.-Z. Song, Y. Wang, X.-Y. Li, and O. Frieder. Localized algorithms for energy efficient topology in wireless sensor networks. In *MobiHoc*, 2004.

[20] M. Youssef, M. Younis, and K. Arisha. A constrained shortes-path energy-aware routing algorithm for wireless sensor networks. In *IEEE-WCNC*, 2002.

[21] V. Zadorozhny, P. Chrysanthis, and A. Labrinidis. Algebraic optimization of data delivery patterns in mobile sensor networks. In *MDDS Workshop*, 2004.

[22] F. Zhao and L. Guibas. *Wireless Sensor Networks: an Information Processing Approach*. Morgan Kauffman, 2004.

[5] [8] presents 18 reasons why long hops may be preferred to small ones.