The Computer Journal Advance Access published April 9, 2010

doi:10.1093/comjnl/bxq034

Controlled Multi-Path Routing in Sensor Networks Using Bezier Curves

OLIVIU GHICA, GOCE TRAJCEVSKI*, PETER SCHEUERMANN, NIKOLAY VALTCHANOV AND ZACHARY BISCHOF

Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60208, USA

**Corresponding author: goce@eecs.northwestern.edu, g-trajcevski@northwestern.edu*

We address the problem of extending the lifetime of wireless sensor networks using multi-path routing based on a family of flexible routes with soft quality of service guarantees in terms of the packets' delivery latency. We introduce a methodology based on Bezier curves as guiding trajectories in the routing process and we address the balancing of the workload among neighboring nodes. An added benefit, due to the flexibility of the Bezier curves, is that the shapes of the (alternate) routes can be constructed in a manner that prolongs the lifetime of the nodes in the vicinity of a given source/sink. We describe a forwarding algorithm, where the relay nodes can determine locally the Bezier curve they belong to and which requires only the transmission of the so-called control points that determine the shape of one (boundary) curve. We also show how our forwarding algorithm can be adapted to incorporate the sleep-schedule of the individual nodes, thereby further prolonging the networks' lifetime. Our simulations demonstrate that the Bezier-based routing algorithms can yield significant improvements in the networks' overall lifetime.

Keywords: wireless sensor networks; multi-path routing; Bezier curves; network lifetime

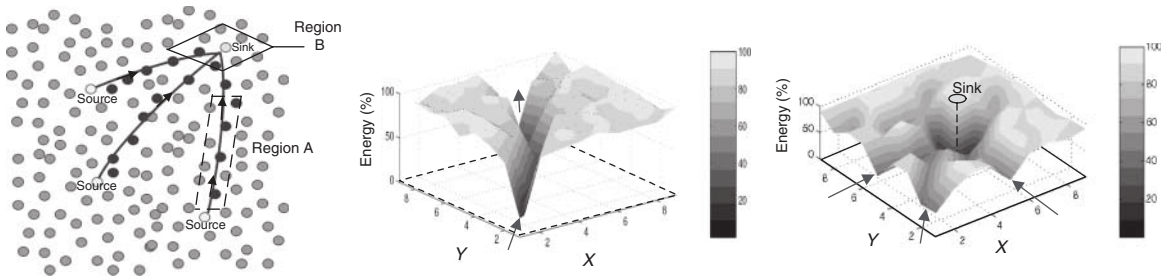
Received 22 January 2010; revised 22 January 2010
Handling editor: Yu-Chee Tseng

1. INTRODUCTION AND MOTIVATION

A wireless sensor network (WSN) represents a distributed monitoring environment which typically consists of a large collection of *sensor nodes*, that is, devices characterized by limited energy resources, computational power, memory space and communication capabilities [1, 2], that are able to form a communication network in an ad hoc manner and coordinate their activities in order to achieve a particular task. In order for the WSN to provide a satisfactory level of a Quality of Service (QoS) (e.g. in terms of data latency, accuracy etc.) for a particular task, careful algorithmic designs are needed at the application, routing and media access layer [3]. However, given the limited power-resources of the individual nodes, an important parameter is the time-extent during which a WSN is operational, namely, the network's *lifetime* [4, 5].

There is no universal definition of the *lifetime* of a WSN and, typically, it is dictated by the specifics of the application of a given WSN. Several lifetime definitions have been given in [4, 5], such as the interval of time until a certain percentage of the network's nodes fail to operate, or the interval of time during which the network maintains connectivity. Various techniques have been proposed to reduce the energy-costs associated with particular network tasks, e.g. in-network data aggregation [6], filtering [7], compression [8] and optimal-path routing [9]—each one considering the energy-efficiency aspect as part of the lifetime maximization problem. However, it has been recognized that lifetime maximization and energy-consumption minimization are correlated but distinct problems [4, 5] and, in addition, it has already been demonstrated that, in general settings, the network lifetime optimization problems are NP hard (cf. [10]).

The goal of this work is the development of a novel multi-routing approach that aims at extending the lifetime of a WSN while ensuring soft guarantees regarding the timely delivery of data-packets. In typical scenarios, long-term continuous queries require periodic sampling/readings of the desired values from a set of nodes (sources) and their transmission to the nodes where these particular queries originate (sinks). In the simplest form of a continuous query, a sink node requires readings from a single source, which may be either the sole producer of the



(a) Scenario illustrating three source-to-sink data-streams

(b) In-network energy distribution Region A

(c) Energy distribution in the vicinity of the sink node Region B

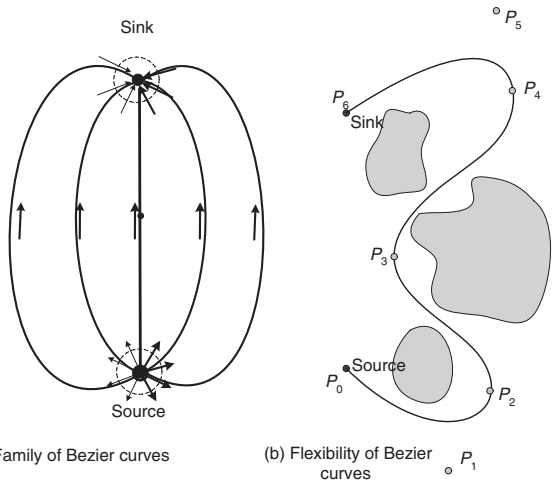
FIGURE 1. The effect of long-term queries over the energy distribution in a sensor network.

data, or represent aggregate values from a region of interest. Such queries tend to exploit a *subset* of the nodes in the network which, in turn, may yield a large discrepancy in the distribution of the overall available energy which, in extreme cases, may even cause disconnectivity. Due to the disconnectivity, the network's lifetime may be rendered as 'expired', although there may be a significant amount of energy left in its constituent nodes.

The scenario depicted in Fig. 1a illustrates three long-term queries dispatched in a WSN, for which three distinct routes are constructed for transmitting the data-streams toward the common sink. Figure 1b illustrates part of the effects over time of the long-lived data-streams routing in the 'transit' region *A* of the network, namely an energy 'gap' that developed due to extended usage of a subset of relay nodes located in that area. As mentioned, this may lead to the appearance of holes in the network and loss of connectivity. Energy depletion is likely to be even more severe in the vicinity of the sink node, as illustrated in Fig. 1c. When multiple routes are related to a single sink, the nodes in its vicinity tend to deplete at a higher rate than other, more distant nodes due to their increased utilization, since they may need to serve more than one data-stream. In addition, those nodes are subject to increased packet collisions and subsequent retransmissions due to the spatial proximity of the routes from different data-streams [11, 12].

The main motivation for this work is based on the observation that, typically, when long-running queries are executed, the nodes in the proximity of a designated route maintain higher energy reserves than the ones involved in servicing that particular route. Hence, if one could use these otherwise dormant nodes for routing purposes, then the variance of the energy levels in the proximity of the designated route over time will be reduced. Moreover, if carefully used, routing along these alternative nodes will not add significant delay to the packet delivery, when compared with the original route. Using multiple paths for routing in WSN settings has already been investigated from a 2-fold perspective: (i) load-balancing which, in turn, prolongs the lifetime of the network and (ii) robustness of the transmission [13–19]. In this sense, our work belongs to the first category—balancing the loads of the participating nodes, for the purpose of minimizing the energy-level discrepancies among the neighboring nodes, while providing a soft QoS guarantees in terms of the latency of delivering the sensed data.

Specifically, in this work we propose to construct multiple routes based on Bezier curves [20, 21], which have been extensively studied in graphics and CAD/CAM applications. One of the most important properties of the Bezier curves is that they have a high degree of flexibility in the construction of their shape, which can be achieved with a set of *control points*. As an example, Fig. 2a illustrates a family of four Bezier curves that can be used as alternate routes for a given (*sink, source*) pair, in addition to the shortest path, i.e. the direct line segment between the source and the sink. Each of these curves is actually a rational polynomial of a third degree, it requires only four control points to construct it. More importantly, the coordinates of those four control points are the only overhead in terms of transmission cost, because every sensor node can decide locally, in a distributed manner, whether it should participate in the transmission along a particular route or not. As we will demonstrate, the construction of the entire family can be done based on the set of initial locations of



(a) Family of Bezier curves

(b) Flexibility of Bezier curves

FIGURE 2. Motivation: routing with Bezier curves.

the control points determining the outermost boundary, that is, the route with the longest delay, plus a few other parameters. Regarding our observation about the utilization of the nodes in the vicinity of the source/sink nodes, Fig. 2a shows how one can utilize a larger area of the nodes on those regions by a careful selection of the control points. To further illustrate the flexibility of the Bezier curves, Fig. 2b shows how, in the case where the locations and shapes of existing holes are known, one can construct a route based on Bezier curves with seven control points ($P_0, \ldots, P_6$) that bypasses them. Observe that it would not be possible to construct such a route.

This work builds upon the results presented in [22], by introducing the concept of *stream-pipes* constructed around the Bezier curves of a given family, for the purpose of forwarding node selection. In addition, we address various run-time aspects and present a detailed performance evaluation of the proposed methodologies. Specifically, the main contributions of this work can be summarized as follows:

(i) We present a methodology which, in response to a request for data values from a given source, creates a family of routes anchored at the (*source, sink*) nodes. These can be used to perform an alternating multi-path routing of data-packets, in a controlled manner, while ensuring soft QoS guarantees in terms of bounded end-to-end communication delay. Since each route is based on Bezier curves, we present a distributed algorithm that performs the mapping of locations along the given route into the actual sensor nodes, in order to enable a trajectory-based forwarding (TBF) [23] of the data packets. An important added benefit of our approach is that we can manipulate the area (equivalently, balance the load) of the active nodes in the vicinity of the source and the sink.

(ii) We extend our routing methodology, in the sense that it not only alternates among different routes, but it also considers *inter-route* alternation among the nodes in the vicinity of a particular route. In addition, we exploit the capability of coordinating the sleep-scheduling among the nodes from different routes.

(iii) We provide an experimental evaluation of the benefits of our proposed techniques, which demonstrate that significant extensions of the lifetime can be achieved while incurring a small increase of the overall energy consumption.

The rest of this article is organized as follows. In Section 2, we present the necessary background. Section 3 describes how a family of alternating routes can be generated based on Bezier curves. Section 4 introduces the fundamentals of multi-path routing using 'stream-pipes' and presents policies for alternating among established pipes. In Section 5, we address some system-wide aspects of the real-time awareness of the Bezier-based routing model. In Section 6, we present the experimental evaluation of the proposed routing techniques. Section 7 positions our work with respect to the related literature. Finally, Section 8 concludes the article and Section 9 outlines directions for future work.

2. PRELIMINARIES

We assume settings in which we are given a set $\mathrm{SN} = \{\mathrm{sn}_1, \mathrm{sn}_2, \ldots, \mathrm{sn}_N\}$ of N sensor nodes that are deployed over a given area of interest. Each node $\mathrm{sn}_i \in \mathrm{SN}$ has a unique, fixed physical location in the 2D geographic space, represented as a pair (x_i, y_i) corresponding to the X and Y coordinates in the reference system. We assume that each node is capable of determining its location (x_i, y_i) at run-time, either by means of location hardware, such as a GPS device, which, for example, is present on the MTS420CA Mica Mote board from Crossbow Technology Inc, or by implementing a location discovery algorithm [24–27].

Each node is also equipped with a small omnidirectional radio device that can be used to establish communication links with other nodes. Let R denote the communication range of each radio device. In practice, the effective communication range is smaller than R due to various medium-perturbations, such as physical obstacles or interference. Hence, we denote as $R^* = \tau R$, where $\tau \in (0, 1]$, the *effective* communication range of a sensor node. The parameter τ may be obtained by analyzing the average one-hop length of the communication links in the network. Due to the limited spatial coverage of the radio device, each node $\mathrm{sn}_i \in \mathrm{SN}$ can communicate directly with only a subset of nodes from the network, which form its set of *neighbors* (with known locations) $\mathrm{NB}(\mathrm{sn}_i) = \{\mathrm{sn}_j |\ \mathrm{d}(\mathrm{sn}_i, \mathrm{sn}_j) \leq R^*, i \neq j, \mathrm{sn}_j \in \mathrm{SN}\}$, where $\mathrm{d}(\mathrm{sn}_i, \mathrm{sn}_j)$ represents the Euclidean distance $\|(x_i, y_i) - (x_j, y_j)\|$ between the locations of the two nodes sn_i and sn_j. We also assume that the nodes are behaving in a cooperative manner [28], in the sense that no node will maliciously refuse to forward any packets.

2.1. Bezier curves

The routing methodology presented in this article relies on the concept of *Bezier curves*. Developed by P. Bezier in the 1970s for CAD/CAM operations [21], these curves have been used in many areas of engineering and computer graphics, and there are many algorithms for their generation [20]. Without loss of generality, the following analysis is restricted to the 2D Euclidean space.

In its general form, a Bezier curve is defined over a set of $n+1$ points: $P_0, P_1, \ldots, P_n$, called *control points* and a parameter $u \in [0, 1]$, and is specified using a nth degree polynomial:

$$\overrightarrow{C(u)} = \sum_{i=0}^{i=n} B_{i,n}(u) \cdot \overrightarrow{P_i}, \quad u \in [0, 1], \tag{1}$$

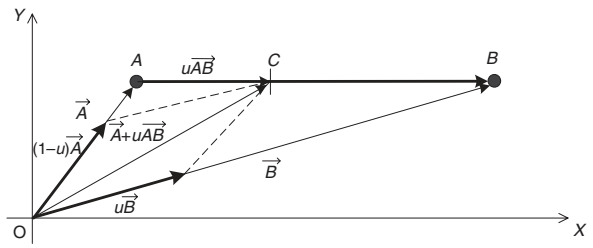


FIGURE 3. Basic step of the de Casteljau algorithm.

where $\overrightarrow{P_i}$ denotes the vector[1] initiating at the origin (0,0), and with a terminus at the 2D point P_i. Each $B_{i,n}(u)$ denotes the so-called *Bernstein polynomials* [20], defined as follows:

$$B_{i,n}(u) = \begin{cases} \binom{n}{i} u^i (1-u)^{n-i} & \text{for } i \in \{0, 1, \ldots, n\}, \\ B_{i,n}(u) = 0, & \text{otherwise} \end{cases} \tag{2}$$

and $\binom{n}{i}$ stands for the standard *binomial coefficient* [29].

Bezier curves are a special kind of the general class of curves called *splines*, used in computer graphics and computer-aided geometric modeling of curves and surfaces [30].

2.1.1. *Construction of Bezier curves*

Given a set of $n+1$ control points that are defining a Bezier curve, it is important to find the actual geometrical representation of the curve in the deployment region of the sensor network, such that individual nodes can evaluate their proximity to the curve. Since Bezier curves are parametric curves, every point pertaining to these curves is 'indexed' by a normalized parameter $u \in [0, 1]$. The mapping is performed through the evaluation of the Bezier polynomial $C(u)$, with $C(0)$ being mapped to one end of the curve, which coincides with P_0, and $C(1)$ to the other end of the curve, which coincides with P_n. The rest of the values $u \in (0, 1)$ map the remaining points of the curve between its two ends. From a practical perspective, however, the direct evaluation of $C(u)$ through Equation (1) is not numerically stable, and a better alternative—the *de Casteljau* [31] algorithm, which uses a recursive method to evaluate the Bernstein polynomials, is often used in practice.

Given a line segment $\overline{AB}$, the fundamental concept of de Casteljau's algorithm is to find a point C on $\overline{AB}$ that divides it into two segments $\overline{AC}$ and $\overline{CB}$ whose ratio is $u/(1-u)$, for a particular $u \in [0, 1]$ (cf. Fig. 3).The vector corresponding to the location of the point C of can be expressed as

$$\overrightarrow{C} = \overrightarrow{A} + u \cdot \overrightarrow{AB} = (1-u) \cdot \overrightarrow{A} + u \cdot \overrightarrow{B}. \tag{3}$$

Based on relation (3), *de Casteljau*'s algorithm is implemented as shown in Algorithm 1 (cf. [31]).

[1]Without loss of generality, we assume the vector space to be the Euclidean 2-space $\mathcal{R}^2$. When there is no ambiguity, we omit the vector ($\rightarrow$) symbol.

Algorithm 1

INPUT: $P = [P_0, P_1, \ldots, P_n]$ an array of $n+1$ control points; $u \in [0, 1]$
OUTPUT: The point on curve, $C(u)$
1. for $(i = 0; i \leq n; i{+}{+})$ **do**
2. $Q[i] = P[i]$; //save input.
3. end for
4. for $(k = 1; k \leq n; k{+}{+})$ **do**
5. **for** $(i = 0; i \leq n-k; i{+}{+})$ **do**
6. $Q[i] = (1-u)\,Q[i] + u \cdot Q[i+1]$
7. **end for**
8. end for
9. return Q[0] // which corresponds to C(u)

Figure 4 illustrates the geometrical interpretation of the *de Casteljau*'s algorithm for a cubic Bezier curve and a specific instance of u (in this example, $u = .3$). In the first iteration, given the line segments $\overline{P_0P_1}$, $\overline{P_1P_2}$ and $\overline{P_2P_3}$ that result from connecting consecutive pairs of control points, the algorithm finds the locations of the points that split these line segments in the $u/(1-u)$ ratio. These split-locations are interpreted as another set of control points that are used in the subsequent iteration. The algorithm stops when there is only one split possible and it guarantees [31] that the location of the last split is also on the curve, which coincides with $C(u = .3)$.

2.1.2. *Properties of Bezier curves*

Bezier curves have a number of important properties that have been identified in the literature [30], which we will use when designing the routing algorithms. Below we discuss some well-known properties:

(i) *Convex hull*—all the points on the curve are contained in the convex hull of the control points. This property is useful when calculating an approximation of the spatial span of a particular Bezier curve.

(ii) *Pseudo-local control*—moving one control point has predictable effects over the changes of the curve's shape and the impact of those effects is larger to the section of the curve in the vicinity of that particular control point. This is illustrated in Fig. 5a, where the original control point P_2 has been moved to a new location—P_2', and the modified Bezier curve is depicted with dashed lines. Observe how the location of the point P from the original curve, corresponding to $C(u)$ for a particular value $u \in [0, 1]$, did not change drastically on the modified curve, where the same value of u generates $C(u)$ corresponding to the point P'. This is due to the fact that P is in the region that is not heavily influenced by the value of the second control point, P_2. This property implies that when local adjustment of a given route are needed, one can manipulate the position of only those control points that most heavily influence the spatial locality in question, knowing that shape of the

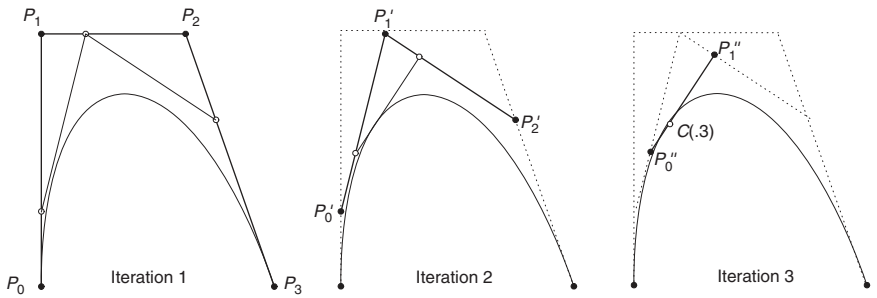


FIGURE 4. Graphical interpretation of de Casteljau's algorithm for $n = 3$ and $u = .3$.

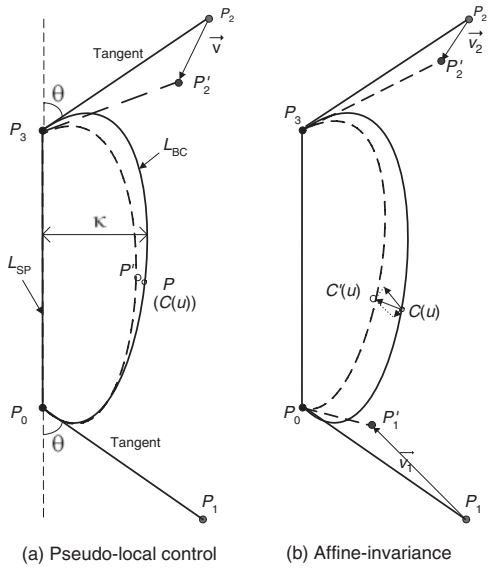


FIGURE 5. Properties of Bezier curves.

curve will not be affected too much in spatially distant locations.

(iii) *End-point interpolation*—the curve always passes through the end-control points P_0 and P_n and, moreover, the segments $\overline{P_0P_1}$ and $\overline{P_{n-1}P_n}$ are tangents to the curve. (cf. Fig. 5). As we will demonstrate, this property enables utilization of the nodes in a wider area around the source/sink.

(iv) *Affine invariance*—in order to apply a particular affine transformation to all the points on a curve, corresponding to all values of u, it suffices to apply the transformation to the control points and apply any standard (e.g. de Casteljau) algorithm using the new set of control points for a given value of u. In general, if the m^{th} control point P_m is translated by a vector $\vec{v}_m$, then for each value of the parameter u, the corresponding point on the curve will be translated by a collinear vector, properly scaled at the m^{th} Bernstein polynomial:

$$C_{\text{new}}(u) = C(u) + B_{m,n}(u) \cdot \vec{v}_m. \tag{4}$$

In particular, as illustrated in Fig. 5b, if the control points P_1 and P_2 corresponding to an initial location of a Bezier curve are translated to new locations P_1' and P_2' by vectors $\vec{v}_1$ and $\vec{v}_2$, respectively, any point P on the curve is moved to its new location P' on the new dashed-line curve. For a given value of u, the effect of the deformation is actually a translation by a vector that is a linear combination $B_{1,3}(u)\vec{v}_1 + B_{2,3}(u)\vec{v}_2$. This property enables a family of routes, based on a given initial Bezier curve, to be constructed locally.

2.2. Approximating a Bezier curve

The methodology that we present in this article relies on a discrete approximation of the Bezier curves, which can be obtained as follows [32]. Let $j \in \{0, 1, 2, \ldots, k\}$ and let $u_j = j\delta$, where $k\delta = 1$, i.e. we discretize the interval $[0, 1]$ into k equal subintervals of size $\delta = 1/k$. Using the values of u_j's only and approximating the rest of the curve by straight line-segments between two consecutive discrete values (e.g. u_j and u_{j+1}), the approximation can be specified as the set of points:

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

where

$$\hat{B}_{i,n}(u) = \begin{cases} B_{i,n}(u) & \text{for } u = u_j, \\ B_{i,n}(u_j) + (u - u_j) & \\ \quad \dfrac{B_{i,n}(u_{j+1}) - B_{i,n}(u_j)}{u_{j+1} - u_j} & \text{for } u \in (u_j, u_{j+1}). \end{cases} \tag{5}$$

Observe that, in order to calculate the corresponding points on the approximated Bezier curve, we actually run *de Casteljau*'s algorithm only on the discrete values of u_j, and the points on the curve whose values are in between two consecutive discretization points (e.g. $\forall u \in (u_j, u_{j+1})$) are approximated using linear interpolation. Hence, instead of the actual Bezier curve, we have a sequence of consecutive line segments

(polyline), whose end-points belong to the Bezier curve obtained by the given control points. Lemma 1 in [32] 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 $(1/\delta^2)n(n-1)\lfloor(n+1)/2\rfloor$ 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 discretization step of the interval [0, 1].

3. BEZIER-BASED ROUTING MODEL

We introduce now the multi-path routing model based on Bezier curves.

We reiterate that our goal is to balance the load among the sensor nodes in a given geographic region, while not exceeding certain QoS constraints. Essentially, we construct a *collection* (family) of routes for a particular request; each individual route relies on TBF, introduced in [23], which is based on the following principle: packets are to be delivered by following as closely as possible a specified trajectory which originates at the source of the data-stream and terminates at the sink. One of the advantages of the TBF-based approach is that no complex routing structures need to be built and maintained globally. Although the goal of the TBF-based routings is to ensure that the data-packets follow a particular trajectory as closely as possible, in reality, the packets may flow within a bounded area around that trajectory, as we shall explain in more detail in Section 4. In the rest of this section, we cast our work—using Bezier curves to construct the family of parametric curves, in the TBF framework. As we will demonstrate, one specific advantage of using Bezier curves is the additional flexibility in terms of exploiting the nodes in the vicinity of the source/sink, when compared with other types of curves such as, for example, ellipses [33].

3.1. Establishing Bezier-based trajectories

The first question that needs to be addressed when constructing a Bezier-based trajectory to be used for packet-forwarding between a given source and sink nodes, concerns the selection of the control points. One can obviously assume that the end-points of a Bezier curve will correspond to the source node (P_0) and to the sink node (P_n).

When it comes to the rest of the control points $P_1, P_2, \ldots, P_{n-1}$, a number of desiderata of long-running queries need to be considered. One important parameter which, in realistic scenarios, is likely to be specified by the user is:

(i) Δ, the maximum admissible end-to-end communication delay of the packets forwarded along any of the existing Bezier trajectories.

Two additional parameters constrained by Δ can be specified as follows:

(i) κ, the admissible distance that the curve should have from the $\overline{P_0P_n}$ which, in a sense, bounds the geographic-deviation of the location of the nodes that can be used for transmitting a given packet. Note that this is, in a sense, proportional to the delay that the transmission of a packet can have with respect to its routing along the shortest path from P_0 to P_n.

(ii) Θ, the allowable 'curvature' in the nearby vicinity of the source and sink nodes, which determines the acceptable locations of the sink/source neighboring nodes that may be used for forwarding purposes.

Observe that the selection of both κ and Θ are aimed at maximizing the number of overall nodes that can be used to relieve the load of the nodes along the shortest path.

The end-to-end communication delay can be linked to the length of a particular Bezier curve which, if we discretize it, can be approximated as follows:

$$\begin{aligned} L_{\mathrm{BC}} &= \sum_{j=1}^{j=k} \overline{C(u_j)C(u_{j-1})} \\ &= \sum_{j=1}^{j=k} \left(\left[\sum_{i=0}^{i=n} \left(B_{i,n}(u_j)P_{ix} - B_{i,n}(u_{j-1})P_{ix} \right) \right]^2 \right. \\ &\qquad \left. + \left[\sum_{i=0}^{i=n} \left(B_{i,n}(u_j)P_{iy} - B_{i,n}(u_{j-1})P_{iy} \right) \right]^2 \right)^{1/2} \end{aligned} \tag{6}$$

Now, considering that the length of the shortest-trajectory path between the source and the sink (L_{SS}) is given by the segment $\overline{P_0P_n}$ as follows:

$$L_{\mathrm{SS}} = \sqrt{(P_0^{(x)} - P_n^{(x)})^2 + (P_0^{(y)} - P_n^{(y)})^2}, \tag{7}$$

where $P_0^{(x)}$, $P_0^{(y)}$, $P_n^{(x)}$, $P_n^{(y)}$ represent the components of P_0 and P_n along the X and Y axes, and assuming that the time delay is proportional to the length of the route, we can estimate the relative increase factor of the end-to-end communication delay along a particular Bezier trajectory (BC) as $L_{\mathrm{BC}}/L_{\mathrm{SS}}$. Consequently, the length of the Bezier curve needs to be restricted to satisfy the constraint:

$$\Delta < \Delta_{\mathrm{SS}} \cdot L_{\mathrm{BC}}/L_{\mathrm{SS}}, \tag{8}$$

where Δ_{SS} is the average delay experienced by data-packets along the shortest trajectory path.

In order to control the degree of the 'curvature' around the end-control points, we refer to the end-point interpolation property of Bezier curves, which states that the segments $\overline{P_0P_1}$ and $\overline{P_{n-1}P_n}$ are tangents to the curve. Thus, we can now define more formally the curvature around the sink (respectively, the source), as the value of the angle formed between $\overline{P_0P_1}$ and

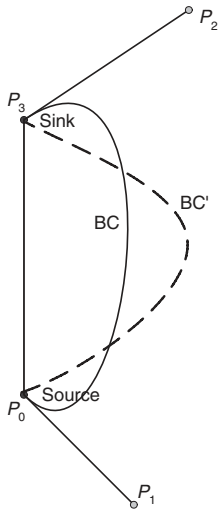


FIGURE 6. Flexibility of the third degree Bezier curves.

$\overline{P_0P_n}$ (respectively, $\overline{P_nP_{n-1}}$ and $\overline{P_0P_n}$), as shown in Fig. 5a (for $n = 3$). Given κ and Θ, we still have a number of degrees of freedom with regard to the actual positions of the control points $P_1, \ldots, P_{n-1}$. We note, however, that κ and Θ are constrained by Δ, which means that, in general, a number of feedback loops may be required in order for the chosen control points to satisfy the end-to-end communication delay. As a specific example, one may be interested in a curve which, for a given Δ, κ and Θ), ensures that the area bounded by the Bezier curve and the line segment $\overline{P_0P_n}$ is maximal. An illustration is provided in Fig. 6, which shows two Bezier curves with four control points (third degree polynomials). The dashed curve (BC′), obtained by using different control points $P_1' \neq P_1$ and $P_2' \neq P_2$, has a wider area-span (in the central region) than the solid curve (BC) and, consequently, it involves more nodes in processing a given request, while retaining the bound on the delay. However, BC′ exploits fewer nodes in the vicinity of P_0 and P_1. This example illustrates that the selection of exact values for the parameters κ, Θ and Δ is an instance of a constraint optimization problem, whose solution is beyond the scope of this work. We note that, in practice, the convex hull of the control points can be used as a first approximation for calculating the values of some of the parameters of interest (e.g. Δ, Θ and κ).

3.2. Building a family of Bezier curves

Given a particular Bezier curve, we can utilize it as a route to relieve the load of the nodes that participate in the optimal (shortest) route between the sink and the source. However, one Bezier curve enables the utilization of only a small subset of nodes between the source and the sink nodes. On the other hand, the degree of lifetime extension, is proportional to the total number of sensor nodes that can share the load and serve on alternate paths. Hence, a larger family of curves between the source and the sink is desirable. In order to construct this family, we start first by determining Δ, the maximum end-to-end communication delay acceptable to the user and require that the length of every Bezier curve satisfies (8). As we will demonstrate shortly, this amounts to constructing two 'boundary' Bezier curves (one to the left and the other to the right of the optimal path) and requiring that all curves in the family fall within these boundaries. In addition, our construction of the alternative Bezier curves will guarantee that they are non-intersecting, as opposed to braided trajectories [14]. The latter exhibit an increase in the collision of packets under high data delivery rates that causes correspondingly an increase in the end-to-end communication delay. We shall describe now our method of obtaining a parametric set of Bezier curves.

Without loss of generality, in the following we will assume that the source and the sink nodes, corresponding to the first (P_0) and the last (P_n) control points are forming a line segment parallel to the Y-axis—otherwise, a proper transformation of the coordinate system can be performed [21]. The first step consists in determining the two 'bounding' Bezier curves that will satisfy, at the limit, the constraints of a long-running query. Let us denote by $\mathrm{BC}_{(R)}^{\max}$ and $\mathrm{BC}_{(L)}^{\max}$ the right-hand side and the left-hand side bounding Bezier curves relative to the shortest-trajectory path $\overline{P_0P_n}$ (cf. Fig. 7). These bounding Bezier curves, will form an 'envelope' for the admissible region that may be used for forwarding purposes. Next, we generate an entire set of curves by relying on the affine invariance property of the Bezier

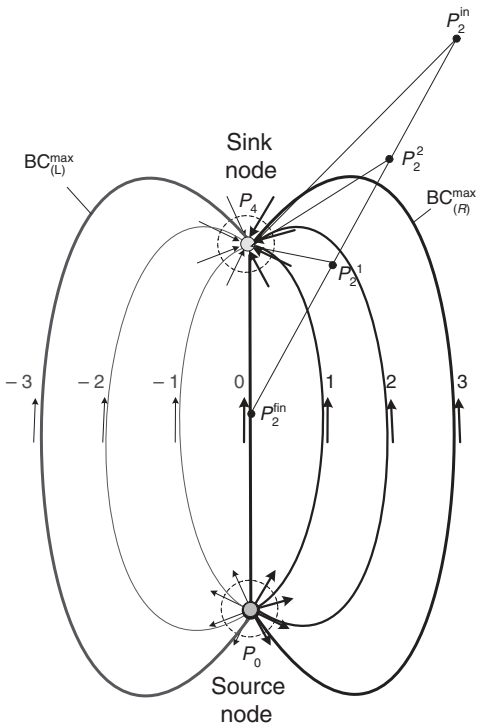


FIGURE 7. Generating and indexing a family of Bezier curves.

curves (cf. Section 2). For simplicity, we focus on explaining how to generate the collection of curves based on the control points of one of the two bounding Bezier curves, say, ($BC_{(R)}^{max}$). In order to select a set of discrete points which will be used as control points for the alternate routes, we proceed as follows:

(i) Let P_i^{in} denote the initial location of the ith control point of one of the bounding Bezier curves.
(ii) Select a set of points on the shortest trajectory-path segment $\overline{P_0P_n}$ as 'final' control points (P_i^{fin}).
(iii) The control points P_i of the additional Bezier curves that will define the family of alternative curves will be located on the segment $\overline{P_i^{in}P_i^{fin}}$.

The number of curves in the family is determined by the selection of a *discretization interval* $1/\lambda$ that will split $\overline{P_i^{in}P_i^{fin}}$ in λ equally-sized segments $\overline{P_i^0P_i^1}, \ldots, \overline{P_i^{\lambda-1}P_i^{\lambda}}$, where $P_i^0 = P_i^{fin}$ and $P_i^{\lambda} = P_i^{in}$. Each discrete point P_i^j, $0 \leq j \leq \lambda$ represents the ith control point of the jth Bezier curve, relative to the corresponding boundary $BC_{(R)}^{max}$. Clearly, this procedure needs to be repeated for the other bounding Bezier curve (in this case $BC_{(L)}^{max}$), and the resulting family is given by the union of the families generated by the two bounding Bezier curves. The family of alternative routes will be indexed through the set $\{-\lambda, -(\lambda-1), \ldots, -1, 0, 1, \ldots, (\lambda-1), \lambda\}$, as illustrated in Fig. 7, for the case of a cubic Bezier curve and $\lambda = 3$. Thus, $2 \cdot \lambda + 1$ represents the cardinality of the family of alternative routes situated on both sides of $\overline{P_0P_n}$.

An important observation is that, upon their deployment, the locations of the individual sensor nodes need not fall exactly on a particular Bezier curve. Hence, each node needs to be able to determine which particular route it is serving. We will address this issue in greater detail in Section 4, however, we mention it here to illustrate some factors that can influence the selection of λ. If λ is too large ($1/\lambda$ small), then consecutive Bezier curves will be too close to each other and a particular sensor node may serve one route defined by more than one curve. On the other hand, if λ is small, the generated family of curves may cover too few of the nodes located between the bounding Bezier curves, and thus reduce the effectiveness of the intended load balancing.

An additional improvement in performance can be achieved by a judicious selection of the P_i^{fin}, $i \in \{1, \ldots, n-1\}$ along the segment $\overline{P_0P_n}$, which affects the workload of the nodes in the vicinity of the source/sink. Figure 8 illustrates various strategies for the selection of the final control points. It presents three different families of cubic Bezier curves based on the same bounding Bezier curve, which differ in the choices of the final positions of the control points. In Fig. 8a, the positions of P_1^{fin} and P_2^{fin} coincide with the mid-point of the segment $\overline{P_0P_3}$; in Fig. 8b, P_1^{fin} and P_2^{fin} are equidistantly distributed along the $\overline{P_0P_3}$ segment; and in Fig. 8c, the positions of P_1^{fin} and P_2^{fin} coincide with P_0 and P_3, respectively (the discretization parameter is $\lambda = 3$). Observe that, even though the families of curves depicted in Fig. 8a–c are very similar, there are important differences toward the end-points of the curves, corresponding to the source and the sink—the changes in the incident angles are different in each case. The case depicted in Fig. 8c represents the best selection of the final control points from the perspective of utilization of a large number of nodes in the vicinity of the sink/source for the entire family of routes. The main reason for this is that the value of Θ is kept constant among all the curves from the collection. In contrast, the family of curves from Fig. 8a yields a worse workload distribution in the one-hop neighboring areas of the source/sink nodes, that is, progressively smaller percentages of the nodes in the corresponding $2\pi - 2\Theta$ sectors participate in the forwarding for different curves, as the value of Θ increases with the proximity of a particular curve to the segment $\overline{P_0P_3}$. Finally, Fig. 8b corresponds to a configuration that achieves a degree of workload distribution somewhere in between the cases depicted in Fig. 8a and c. For comparison, Fig. 8d. illustrates how the neighboring nodes around the source/sink have much lower utilization when an ellipse is used as a base for the construction of routing trajectories.

4. STREAM-PIPES FORWARDING MODEL

In this section, we address the main issues involved in forwarding data-packets along an individual trajectory from a given family of Bezier curves. Toward this, we present a forwarding model based on 'stream-pipes', which is a collection of nodes in a closed spatial proximity of a particular Bezier curve. We define the concept of a stream-pipe and present the forwarding algorithm executed *locally* by an arbitrary node. Subsequently, we focus on the problem of load-balancing within a given stream-pipe and address in detail the behavior of the source.

Whenever a new request is to be posed to the network, its processing requires two main actions: (i) propagating it from the sink to the source, and (ii) coordinating the data sampling and delivery throughout the desired lifetime of the request. From this perspective, there are three main functional categories of the nodes:

(i) sink;
(ii) source;
(iii) relay nodes along a given curve.

We note that the selection of the actual source node is done based on a minor modification of the TBF approach [23]. Namely, the sink generates an *initial request* specifying the geographic location from which the readings are desired. As the request-packet is being propagated, each node checks whether its distance from the destination which is smaller than any one of its neighbors, in which case, it declares itself to be the source. Otherwise, it forwards the packet toward the location initially specified by the sink, in a TBF-like manner.

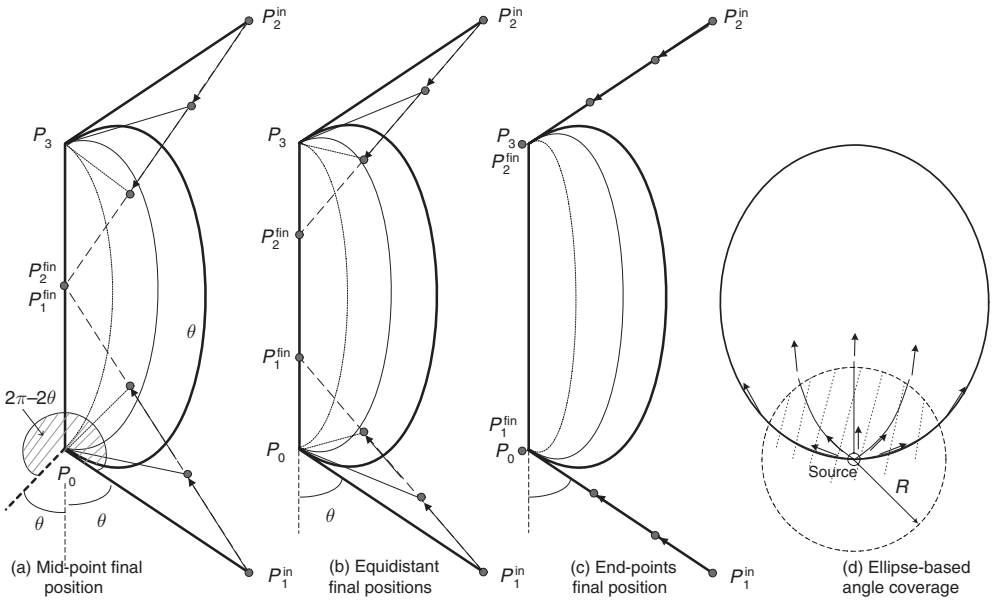


FIGURE 8. Possible selections of the final positions of control points.

4.1. Stream-pipes based on Bezier curves

We proceed now to define in detail the major concepts of stream-pipes, namely, how the relay nodes perform geodesic forwarding, as well as how to determine the number of curves in a stream-pipe and their boundaries.

Given a Bezier curve BC_k, the *stream-pipe* SP_k defined by that curve is the spatial region that consists of all the points that are no further than R^* from BC_k— i.e. the width of the pipe is equal to $2R^*$, the diameter of the effective communication range of each node. An alternative definition of the stream-pipe SP_k is that it is the subset of SN (the set of all the nodes) which consists of the nodes whose locations are inside the region $BC_k \oplus R^*$, where $\oplus$ denotes the *Minkowski Sum* [34] of BC_k and the disk with radius R^*.

We note that, for each stream-pipe SP_i, a *direction* of flow of the packets needs to be defined, which is expected to be 'outbound' from the source and 'inbound' to the sink. However, this intuitive desideratum, when making a forwarding decision, cannot simply use the Euclidean distance between the location of a given node and the location of the sink (respectively, source). As illustrated in Fig. 9c, if the source uses the Euclidean distance as a criterion for forwarding, it will (incorrectly) send the packet to the node 'C' instead of 'B', since 'C' is the node that is furthest from the source itself, and closest to the sink. However, 'B' is the one that is closest *along* the Bezier curve defining that stream-pipe. Similarly, even the node 'B' may prefer 'C' to 'A', if the Euclidean distance was used. Hence, the correct criterion of advancing/forwarding a given packet among the nodes within a given stream-pipe SP_i is to ensure that its *geodesic distance*, i.e. the length of the path traveled by that packet along the nodes within SP_i, is increasing from the sink and decreasing toward the source. On the basis of properties of Bezier curves, we can now formally define the desideratum for geodesic distance. Given two nodes, sn_p and sn_q, we say that sn_p is *further along* BC_i from the source (resp. toward the sink) than sn_q if and only if the value u_p (of the parameter u) used for calculating the location of the point on BC_i that is closest to sn_p is larger than the value of u_q (of the parameter u) used for calculating the location of the point on BC_i that is closest to sn_q. The definition of the Bezier curves (cf. Equation (1)) and the condition $u_p > u_q$ ensure that the geodesic distance from the source along BC_i, is *monotonically increasing* (Fig. 9c). We note that a node sn_p may have more than one 'nearest-neighbor' on a particular curve BC_i, in which case a tie-breaker is needed. In our implementation (cf. Section 6) we have adopted the policy of selecting the largest u-value that is no smaller than the values used by the last node in the stream-pipe.

Another important aspect of designing the stream-pipes, related to determining the actual number of the Bezier curves in the family, is how to determine their *width*. Once again, there are conflicting requirements: (i) in order to maximize the 'spatial utilization', one needs to have *more* and *wider* pipes and (ii) in order to avoid inter-pipe interference, one needs to ensure some minimal distance between the pipes corresponding to neighboring curves from the family. Let $D^{SP}_{i,(i+1)}$ denote the maximal Euclidean distance between the two stream-pipes defined by neighboring Bezier curves $BC_i(u)$ and $BC_{(i+1)}(u)$ for a given $u \in [0,1]$, and let d_i denote the width of SP_i. An upper bound on the value of λ, which determines the total number of the curves in a given family, can be specified as:

$$\lambda \leq \frac{\kappa_{\max}}{(d_i + D^{SP}_{i,(i+1)})}, \tag{9}$$

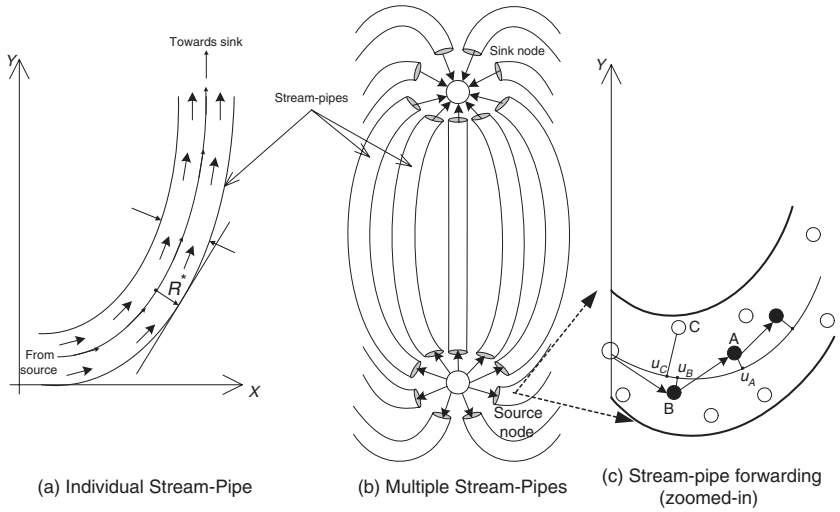


FIGURE 9. Stream-pipes model.

where $\kappa_{\max}$ denotes the largest spatial deviation that a Bezier curve can have from $\overline{P_0 P_n}$.

Clearly, the actual value of λ will depend on the selections of $D^{\mathrm{SP}}_{i,(i+1)}$ and d_i. However, regardless of what those values are (e.g. even if one allows a larger distance between two consecutive stream-pipes), the proximity among the nodes in the vicinity of the source and the sink may not permit such a distribution, as opposed to the nodes, say, in the middle of a given curve. Depending on the actual deployment, this observation remains valid even if $D^{\mathrm{SP}}_{i,(i+1)}$ and d_i are made functions of u. We note that in our implementation, we used the values of $d_i = 2 \cdot R^*$ and $D^{\mathrm{SP}}_{i,(i+1)} = R^*$.

4.1.1. The behavior of relay nodes

Upon receiving a data-packet that needs to be forwarded to the sink, each node must generate an *anchor point*—a 2D point along the respective Bezier curve which determines the corresponding route—in order to decide which one of his one-hop neighbors should be used next in the forwarding process. The generation of the anchor point is based on the parameters of the underlying curve defining the stream-pipe to which a given node belongs, and its location is transmitted along with the regular data-packets. Given a fixed, known anchor point, the forwarding decision is simply based on a greedy choice of a node from the set of neighboring nodes, that is physically closest to that anchor point. As an example, in Fig. 10a the black-filled dots represent the set of anchor points along a particular Bezier curve, and the white disks linked with line segments to them represent the actual forwarding nodes associated with these anchor points. In order to complete the forwarding between the source and the sink, a sequence of such anchor points must be formed in a distributed manner.

On the one hand, each anchor point can correspond to the longest one-hop of communication distance, and hence the physical distance between two consecutive anchor points along a particular route should be equal to the communication range R^*. Using a small distance value for the consecutive anchor points may not only increase the number of hops required to forward a data-packet between the source and the sink node, but also possibly create local cycles. This is illustrated in Fig. 10b. However, enforcing a strict monotonically increasing policy on the geodesic distance of consecutive forwarding nodes will prevent the occurrence of such cycles. On the other hand, a large distance between consecutive anchor points (cf. Fig. 11a) may yield a sequence of forwarding nodes whose positions will not closely follow the prescribed trajectory, as illustrated in Fig. 11b.

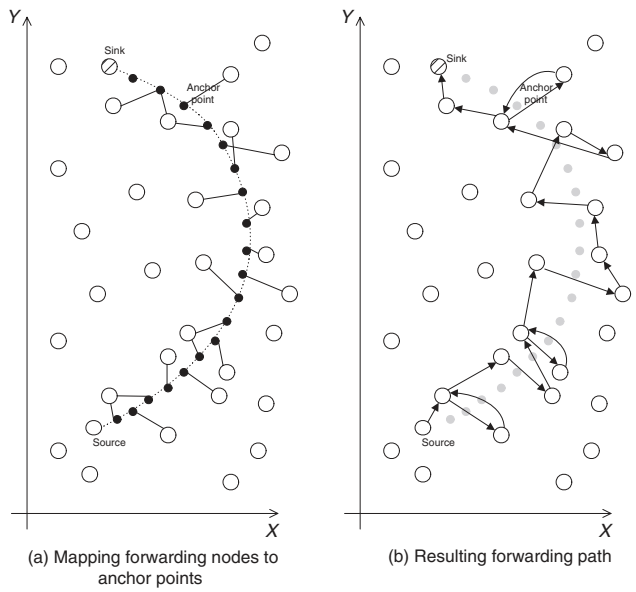


FIGURE 10. Impact of 'small' discretization step on the actual routes.

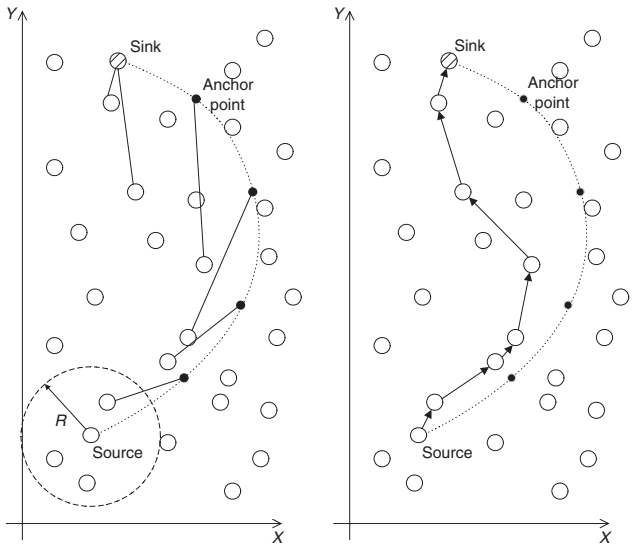


FIGURE 11. Impact of 'large' discretization step on the actual routes.

An important observation is that different Bezier curves from a given family will have a different length and, consequently, a different number of anchor points. The number of anchor points for a particular curve BC_i, denoted by k_i, can be calculated as follows. We divide its approximate length LC_{BC_i}, obtained using Equation (6) in Section 3.1, by R^*, and we let $k_i = \lfloor LC_{BC_i}/R^* \rfloor$. For $i \in \{-\lambda, \ldots, -1, 0, 1, \ldots, \lambda\}$, the value of k_i is used to determine the discretization value for the parameter $u \in [0, 1]$ (cf. Section 2.2). In other words, the discretization step δ_i for approximating the ith Bezier curve with a polyline is obtained as $\delta_i = 1/k_i$. Recall that, for each BC_i, the locations of its control points can be derived using the control points of its *bounding* Bezier curve and their final locations along the segment $\overline{P_0P_n}$. Specifically, the location of the jth control point for the ith curve of the family of Bezier curves, P_j^i is given by

$$P_j^i = P_j^{\text{in}} \pm |P_j^{\text{fin}} - P_j^{\text{in}}| \cdot (i/\lambda), \tag{10}$$

where the selection of '$-$' or '$+$' sign is determined based on whether the ith curve is indexed by a positive ('$-$') or a negative ('$+$') value of its index (i.e. whether the curve is on the right-hand side or the left-hand side of $\overline{P_0P_n}$). Using the corresponding control points P_j^i and the corresponding discretization step of the unit-interval δ_i, the location of the lth anchor point on the ith Bezier curve can be calculated as

$$\hat{C}_i(l \cdot \delta_i) = \sum_{j=0}^{j=n} \hat{B_{j,n}}(l \cdot \delta_i) \cdot P_j^i. \tag{11}$$

Observe that in Equation (11), the values of $\hat{C}_i(l \cdot \delta_i)$ (respectively, $\hat{B_{j,n}}(l \cdot \delta_i)$) coincide with the actual values of $C_i(l \cdot \delta_i)$ (respectively, $B_{j,n}(l \cdot \delta_i)$), computed using Equation (5) (cf. Section 2.2) with the discrete set of values for the parameter u corresponding to $l \cdot \delta_i$. Given the information about the particular curve BC_i (its control points) and the discretization interval for the parameter u on that curve, an individual node, say sn_r, can use Equations (10) and 11, to determine its relative position with respect to a particular anchor point, say, AP_i^j on that curve. More importantly, sn_r can determine the relative position of each of his one-hop neighbors $NB(sn_r)$ with respect to that curve and its anchor points. Instead of having to evaluate an uncountably infinite set of possible values for u, once again sn_r will use the discrete approximation of the particular Bezier curve where the discretization step δ_i is determined based on the stream-pipe SP_i (equivalently, the Bezier curve BC_i) that is closest to sn_r. In order to estimate with a bounded error the distance (of itself or any node from $NB(sn_r)$) to the curve BC_i, the node sn_r will use the distance to the polyline approximating BC_i, i.e. the line segment between the two closest anchor points (cf. Equation (5)).

Algorithm 2 summarizes the basic forwarding methodology using the ideas presented so far. In order to perform the forwarding decision, a node needs to receive the following *control* information within a particular packet Π:

(i) ID, the packet ID (determined by the source).
(ii) sn_{sink}, the ID (respectively, location) of the sink node.
(iii) sn_{send}, the ID (respectively, location) of the node that sent the packet.
(iv) i, the index of the pipe SPi that is currently being used, $i \in \{-\lambda, \ldots, -1, 0, 1, \ldots, \lambda\}$
(v) $P_0^i, P_1^i, \ldots, P_n^i$, the control points of the BCi currently being used, calculated by the source node once for a given $i \in \{-\lambda, \ldots, -1, 0, 1, \ldots, \lambda\}$.
(vi) δ_i, the normalized ($0 \leq \delta_i \leq 1$) discretization step of the ith curve, calculated as $1/\lfloor LC_{BC_i}/R^* \rfloor$.
(vii) the current parameter value of the anchor point used to select sn_r as the receiver of the packet, calculated using $u_l = \delta_i \cdot l$.

In the forwarding algorithm, the receiving sensor node sn_r first checks whether the sink is within its effective communication range and, if so, forwards the data portion of the packet Π directly to it. Otherwise, sn_r determines a new anchor point $AP(sn_r)$ along the stream-pipe SPi and, from the set of its one-hop neighbors $NB(sn_r)$ it selects the one that is closest to the $AP(sn_r)$, to which the data-packet Π with the updated control information is forwarded.

Algorithm 2 *Forward*(Π, sn_r): the forwarding of packet Π by the relay node sn_r
INPUT: $\Pi = ([ID, sn_{sink}, sn_{send}\, i, \delta_i, u_l, P_0^{(i)}, \ldots, P_n^{(i)}]$, data)
$NB(sn_r)$—the set of 1-hop neighbors of node sn_r
OUTPUT: Next-hop neighbor for forwarding the packet toward the sink;
Updated packet Π
1. IF $sn_{sink} \in NB(sn_r)$

2. Send $\Pi.data$ to sn_{sink};
3. **ELSE** // Determine the next anchor point on the curve indexed by i
4. $\text{AP}(\text{sn}_r) = \hat{C}_i(u_l + \delta_i)$ // evaluated using de Casteljau's Algorithm with control points $P_0^{(i)}, \ldots, P_n^{(i)}$;
// Determine the neighboring node to forward Π to
5. Select node $\text{sn}_{\text{next}} \in \text{NB}(\text{sn}_r)$ such that sn_{next} is closest to $\text{AP}(\text{sn}_r)$;
// In case of a tie, select the one furthest from sn_r itself
6. **IF** sn_{next} has been successfully communicated (ACK-ed)
7. // Update Π's relay information:
$\text{sn}_{\text{send}} = \text{sn}_r$ and $u_l = u_l + \delta_i$
8. Send Π to sn_{next}
9. **ELSE**
10. Notify sn_{send} and drop Π
11. **ENDIF**
12. ENDIF

Algorithm 2 is the basic forwarding algorithm executed by a node along a given route, and its complexity is linear in the cardinality of the set of one-hop neighbors of sn_r. In other words, if $|\text{NB}(\text{sn}_r)| = n_r$, the time complexity of Algorithm 2 is $O(\text{DC}(n) \cdot n_r))$, where $\text{DC}(n) = O(n^2)$ is the complexity of executing the De Casteljau's algorithm for BC_i defined via n control points. We note that a particular relay node sn_r in a given stream-pipe needs to execute Algorithm 2 for forwarding only the first time it is used for routing. For subsequent packets, sn_r needs to check whether the relay parameters in Π have been received before, which is the input parameters, the updated relay parameters, as well as the state parameters consisting of sn_{next}, the values of its anchor point and the current route, and the distances of each node in $\text{NB}(\text{sn}_r)$ to the anchor point. Additionally, if sn_r cannot forward the packet Π toward the sink along BC_i, it will notify the sender sn_{send}, as this may indicate a topological hole in the network [35], which will have to be bypassed (if possible) in order to successfully transmit the requested data from the source to the sink.

4.2. Load-balancing in stream-pipes

Although the concept of stream-pipes enables more nodes in the vicinity of a given Bezier curve to be used for routing, this alleviates only partially the problem of workload distribution. We observe that the association of actual nodes to anchor points is *static*, in the sense that an invariant subset of nodes that are mapped to a particular pipe will be actively used, whereas the others inside that same pipe will be idle. This may increase the discrepancy of the energy levels among the nodes within an individual stream-pipe. Similar issues have been observed for shortest path-based routing (cf. [36]) and the proposed solution tried to dynamically adapt the selection of the forwarding nodes along that path. In our settings, instead of a shortest path we have a Bezier curve that is discretized by using a sequence of anchor points, and we allow the nodes in the vicinity of a given curve to participate in the routing based on their proximity to a given anchor point. In order to better exploit the nodes in a given stream-pipe and balance the load among them, we allow for a number of *virtual channels* in the stream-pipe, each one using a different sequence of anchor points along the respective Bezier curve. We achieve it by using a *randomized offset* that is determined by the source to vary the location of the first anchor point along a given virtual channel.

Our method for generating virtual channels proceeds as follows. The source node computes a different randomization factor, denoted by $\omega_{i,j} \in (0, 1]$ for determining the location of the first anchor point along the jth virtual channel of SP_i. Specifically, its location is computed by using $u_{ij_1} = \omega_{ij} \cdot \delta_i$ in Equation (11). The subsequent anchor points are computed by each relay node using the given discretization step value δ_i. This achieves the effect of generating a different sequence of anchor-points for each distinct virtual channel. Hence, two data-packets transmitted by the nodes from a given stream-pipe SP_i, using a different value of ω_{ij} and the same value for δ_i, are likely to be forwarded by different sets of nodes. The reason for this is that each value of ω will determine a different start of a discretization sequence for the parameter u which, in turn, will generate a different sequence of anchor points along the (approximated) Bezier curve. Consequently, we have an increased likelihood that a different sequence of forwarding nodes within a given pipe will be used. Figure 12 illustrates two different virtual channels within a single pipe for $\omega = .4$ and $\omega = .8$, illustrating how different nodes are selected as forwarding nodes for different values of ω.

4.2.1. *The behavior of the sink and source*

The main role of the sink node is to properly initiate a particular *request*, for which the basic attributes are: (i) T—the duration of the sampling; (ii) f_{r}—the frequency/rate of the sampling executed by the source and (iii) the *source*—the source node ID,

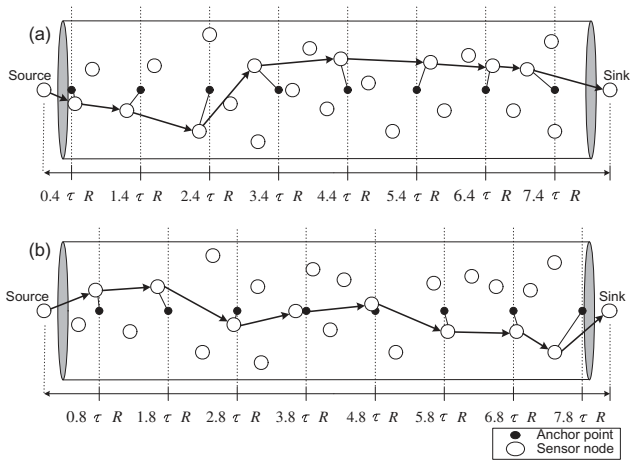


FIGURE 12. Multi-channel routing within a single pipe: a) $\omega = .4$; b)$\omega = .8$.

or equivalently, the physical location from which the readings are requested. An additional responsibility of the sink is to determine the parameters needed for generating the family of curves that will be used for routing purposes, which include the following: (i) the control points of the *bounding* Bezier curves; (ii) the number of alternative routes, i.e. the value of λ and (iii) the final location of the control points. In addition, the sink also determines the value of N_p—the number of consecutive packets that should be sent along an individual stream-pipe, and m_i—the number of virtual channels in the ith stream-pipe SP_i. A REQ (request) packet with this information is sent to the source along the shortest path route, as explained in Section 3.

Throughout the duration T of the sampling period the source node is responsible for collecting the sensed data values, as well as coordinating the delivery of the packets among the $2\lambda + 1$ stream-pipes and among the virtual channels within a single stream-pipe. For that purpose, the source node determines most of the information for obtaining the run-time routing parameters, that are specified as follows:

(i) The index $i \in [-\lambda, \ldots, -1, 0, 1, \ldots, \lambda]$ of the current (Bezier curve of the) pipe P_i.
(ii) The discretization interval δ_i for that pipe.
(iii) The initial value of the parameter u, that is, the value of ω_{i_j} for determining $u_{i_j} = \omega_{i_j} \cdot \delta_i$, which is needed for determining the particular (in this case, jth) virtual channel within a given pipe. Given its knowledge of the one-hop neighbors, the source actually determines a set $\Omega_i = \{\omega_{i_1}, \ldots, \omega_{i_m}\}$ of initial offsets for a given stream pipe SP_i.
(iv) n_c, the number of data-packets that have been sent consecutively through the currently used virtual channel j along SP_i.

Recall that the data stream serving a particular request will be transmitted over $2\lambda + 1$ alternating pipes, however, the relay nodes in each stream-pipe SP_i are divided into virtual channels. Each channel will be responsible for transmitting $\lfloor N_p / \mid \Omega_i \mid \rfloor$ packets, where $\mid \Omega_i \mid$ denotes the cardinality of the set Ω_i. The run-time behavior of the source node is specified by Algorithm 3.

Algorithm 3 Source Routing Behavior
INPUT: R^*, $P_0^{\text{in}}, \ldots, P_n^{\text{in}}$, $P_0^{\text{fin}}, \ldots, P_n^{\text{fin}}$ // received from the sink
STATE-INFO: λ, $\lfloor N_p/m \rfloor$, Ω_i, n_c (counter of sent packets), i (pipe index), j (channel index), ω_{i_j}
OUTPUT: i_{next}, $\delta_{i(\text{next})}$, sn_{next}
along with updated STATE-INFO
1. Initialize:
$i_{\text{next}} \leftarrow i$;
$n_c \leftarrow n_c + 1$;
$j_{(\text{new})} = j$;
// Determine if the virtual channel and/or the stream-pipe need to be updated
2. IF $n_c \geq \lfloor N_p / \mid \Omega_i \mid \rfloor$
3. $n_c = 0$
4. $j_{\text{new}} \leftarrow j + 1$
5. **IF** $j_{\text{new}} > m_i$
6. $j_{\text{new}} = 1$
7. $i_{\text{next}} \leftarrow$ *Select* $i_{\text{next}} \neq i$ from $[-\lambda, \ldots, 0, \ldots \lambda]$ // Switch to a different pipe
8. $\delta_{i(next)} \leftarrow$ calculate the discretization interval/step for $\text{SP}_{i(\text{next})}$
9. **ENDIF**
10. $\omega_{i_{j(\text{new})}} \leftarrow$ get the offset for the j_{new}th channel from Ω_i
11. ENDIF
// Update the STATE-INFO
12. IF $(j_{\text{new}} \neq j)$
13. $u_{\text{next}} \leftarrow \omega_{i_{j(\text{new})}} \cdot \delta_{i(\text{next})}$
14. ENDIF
15. Determine sn_{next} using the forwarding Algorithm 2
16. Compose and send packet Π to node sn_{next}

We observe here that line 7 of the algorithm actually involves a number of computation steps:

(i) Using Equation (10), the source computes the coordinates of each control point of the $i_{(\text{next})}$ pipe.
(ii) The source also computes $\text{LC}_{\text{BC}_{i(\text{next})}}$—the approximate length of the Bezier curve $\text{BC}_{i(\text{next})}$, using Equation (6).
(iii) Lastly, the actual value of $\delta_{i(\text{next})}$ is computed as $\text{LC}_{\text{BC}_{i(\text{next})}} / R^*$.

In addition, the selection of the subsequent stream-pipe in line 7 is something that may affect the energy consumption, which we address in detail in the Section 5.1.

We conclude this section with a few observations regarding the complexity of the Algorithm 3. First, in a similar spirit to Algorithm 2, we note that the source node needs to calculate the respective parameters for routing within a given stream-pipe (e.g. (δ_i, Ω_i)) only once—upon the initial processing of the query request and the very first time each SP_i is used for routing. The calculated values can be stored locally and simply re-used in any subsequent usage of SP_i. If the total number of one-hop neighbors of the source node is n_{src}, then the number of executions of Algorithm 3 as stated above, is $O(n_{\text{src}})$. This worst-case scenario occurs when each one-hop neighbor is used as an anchor point—albeit within different stream-pipe(s). Let n_B denote the combinatorial complexity, i.e. the number of control points of the Bezier curves. Since there are a total of $2\lambda + 1$ different stream-pipes, the time complexity of the initial execution of Algorithm 3 is bounded by $O(n_B^2 \lambda n_{\text{src}})$. Clearly, this cost will be amortized with the subsequent, after initial, uses of each stream-pipe. In addition, as our experiments demonstrate, the amortized cost justifies, in terms of the lifetime criteria, the use of Bezier-based routing as opposed to, for example, k-short—consisting of two line-segments per route, the initial specification of which is just as straightforward as, for example, ellipses (two points/real-valued parameters).

5. SYSTEM-WIDE AND REAL-TIME ISSUES

Thus far, we have presented the basic ideas behind the routing within a particular stream-pipe from the family of Bezier curves. However, there are some other important run-time aspects that need to be considered in order to further prolong the lifetime, decrease the overall energy expenditure of the nodes and ensure the timely delivery of the packets to the sink. In this section, we address these issues in detail.

5.1. Stream-pipe selection policies

Given a collection of stream-pipes, an important decision that the source node needs to make is what *policy* to use for alternating among the pipes when transmitting the sensed data. In Algorithm 3 discussed earlier, this was stated as the choice of the $i(\text{next}) \in \{-\lambda, \ldots, +\lambda\}$. Assuming that the pipes are symmetric around the line segment $\overline{P_0P_n}$ and adopting the counter-clockwise direction of traversing a particular curve from P_0 toward P_n as the positive one, the set of alternating routes can be enumerated as: $S = [-\lambda, -(\lambda - 1), \ldots, -1, 0, 1, \ldots, (\lambda - 1), \lambda]$, corresponding to their left-to-right sequence with $\overline{P_0P_n}$ having the index value of 0.

The policy of alternating among the pipes now amounts to selecting the best permutation of S to be used in transmitting the packets. One may be tempted to use the identity permutation which alternates periodically among the sequence S of pipes, and selects in each period the pipe based on the order in the original sequence S. We call this a *sweep*-based alternating policy among the stream-pipes.

However, as observed in the literature (e.g. [37]), there may be packet-loss due to interference among the nodes along spatially close pipes. When the sampling frequency is high, and a packet is transmitted for every sampled value, spatially close nodes will be employed by adjacent stream-pipes in small time intervals. Specifically, when the *sequential sweep* (modulo λ) is used to select among the candidate pipes, this will result in pipe interference, which is especially observable in the proximity of the sink/source node, under high data rate loads. Motivated by this, we propose the following permutation for alternating among the pipes:

$$S_d = [-\lambda, 1, -(\lambda - 1), 2, \ldots, -2, (\lambda - 1), -1, \lambda, 0]. \quad (12)$$

We call S_d the *constant mid-distance* permutation for alternating among the stream-pipes.

Many other policies are possible, in addition to the sweep-based and the constant mid-distance approaches. As another example, one can *randomly* select which stream-pipe is to be used next. As part of our experiments, we have tested the behavior of *sweep*, *mid-distance* and *random-select* policies. The results are depicted in Fig. 13, which illustrates the energy demands of the participating nodes when transmitting a data-stream between two fixed end-points for a duration of 10 h, every 2 s. The differences in the energy demands are due to the packet collisions occurring as a result of neighboring pipe interference, and subsequent packet retransmissions. As one may observe, the *constant mid-distance* policy is the best performer in terms of energy efficiency. As a consequence, we have adopted it as a stream-pipe selection strategy and used it in our experiments.

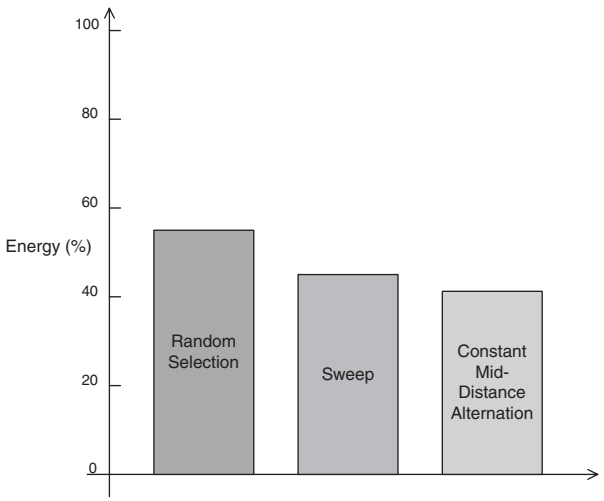


FIGURE 13. Route selection policies.

5.2. Sleep-scheduling of sensor nodes

An additional opportunity to preserve the energy of the nodes is to exploit their capability of turning their radios off, by entering into a *sleep* state [38]. In our settings, at any time only the nodes from one stream-pipe, say SP_i need to be active. Whenever another route is to be selected, the nodes along its stream-pipe SP_j need to be awaken, and the nodes from SP_i may enter the sleep state.

Switching between the sleep/active state is a process that can take from a few microseconds to several hundreds of milliseconds, depending on the type of radio and awakening mechanism a node possesses [38]. As a consequence, the data-packets that are waiting for the transmission at a given sensor node may experience a delay when the next hop is in a sleep state. Additionally, if the frequency by which routes are switched is greater than the maximum frequency needed for each node to switch on/off its circuitry, then one may find nodes that are being mapped to different stream-pipes active simultaneously, even though only one pipe is needed at a time.

Consequently, once a specific stream-pipe is being selected, it may be beneficial to use it for a prolonged period of time before it is replaced; in other words, we use it to send N_p consecutive data packets through it, before selecting another pipe. Ideally, if we consider a query with a life-span of T that requires a data-stream sampling frequency s_f to be delivered across $2\lambda + 1$ alternative pipes, then N_p should be chosen as $N_p = T/(s_f \cdot (2\lambda + 1))$. In this case, the algorithm would access

each stream-pipe from a given family of pipes only once, and use it for an extended period of time. Doing so minimizes the overhead of transmitting and evaluating the Bezier curves as the curve's parameters need to be transmitted only $2\lambda + 1$ times, once for each curve. However, for very large values of T, the energy imbalance may increase significantly among adjacent routes and if we limit the number of packets that can be sent consecutively through each stream-pipe, we can achieve a more even residual energy distribution. As an example, a possible upper bound can be $N_p = Q_{\text{size}}$, where Q_{size} is the capacity of the outgoing queue (buffer) of each sensor node [39]. In this case, the risk of overflow in a pipe, i.e. exceeding the buffer capacity, which would result in a packet drop, of the nodes within is reduced. In our implementation and the experiments, we chose the value of $N_p = \min\{Q_{\text{size}}, T/(s_f \cdot (2\lambda + 1))\}$.

We assume that each node is equipped with a *remotely activated switch* [40] which allows its transceiver to be powered on remotely by a neighboring node, via dedicated paging signals, whenever there is a need for a communication link to be established between the two. In our settings, whenever a particular stream-pipe needs to be used initially (or, after the sensor nodes around it have been put to sleep), a special *WAKE* packet is sent by the source and forwarded along by the corresponding nodes in order to establish the communication channel.

The *WAKE* packet is very similar to the regular packet Π used in Algorithm 2, except it has two specific items:

(i) A dedicated *wake* bit, which is used by the sink node to discard that packet upon receiving. The in-between relay nodes will treat the *WAKE* packet just as a regular data-packet and they will attempt to forward it toward the sink, for the purpose of awakening and establishing the route.
(ii) A dedicated *used* bit which tells the receiving node that (the Bezier curve of that) route, along with the particular channel within the corresponding stream-pipe, have already been used before. Thus, $used = 1$ means that the node sn_r currently receiving the *WAKE* packet has already been given all the information needed to execute the Algorithm 2 and it had the opportunity to store the output sn_{next} from among its neighbors $\text{NB}(\text{sn}_r)$. On the other hand, if $used = 0$, upon receiving the *WAKE* packet, sn_r knows that it has never participated in that particular route/channel, and hence it will have to use (and store for subsequent use) all the parameters from the *WAKE* and execute the Algorithm 2, the output of which will be also be stored for subsequent transmissions.

5.3. Delay budgeting with hole bypassing

Algorithm 2 is designed to operate under the assumption of a densely populated network, where a one-hop neighbor is available for every sensor node and in any desired direction. However, in reality, some nodes or group of nodes may not be able to establish a communication channel. This is the case, for example, when nodes are not physically available in isolated geographic areas (e.g. due to deployment in mountainous and/or hostile environments) or, despite their availability upon the initial deployment, group of nodes in a given region have become unusable (e.g. energy depletion or environmental disasters). Regardless of the reason, the lack of effective availability of nodes in a particular region implies an existence of, so called, *holes* in the network [35, 41].

As we mentioned in Section 3, in the case that the relevant information about the holes, e.g. their location and boundaries, is known beforehand, the sink node may take it into consideration when formulating the request for routing from a given source, and properly construct the Bezier curves that will ensure routes that can bypass them. However, as observed in the literature (e.g. [35]), network-wide management of the knowledge about the holes can be cost-prohibitive, as holes may be created at various points of the network's lifetime, and local routing algorithms are needed that can be used to bypass the hole and relay the packets (as close as possible) to the desired destination.

To cope with this issue, a straightforward modification of the implementation of Algorithm 2 is needed in the steps 5–8 to adopt a GPSR-like behavior [41], that is, selecting the nodes along the boundary of the hole, with a goal of reaching the sink. However, there is an important practical issue that needs to be considered in our settings; namely, when bypassing the hole in a particular reference direction (typically, a counter-clockwise, so that the interior of the hole is always on the left-hand side), the time needed for a packet transmitted along a given Bezier curve BC_i (equivalently, its stream-pipe SP_i) may exceed the acceptable delay bound Δ. Hence, in our current implementation, we have adopted a slightly modified version of the original GPSR approach:

(i) When a hole is encountered, it is bypassed from both directions (clockwise and counter-clockwise).
(ii) Once each of the bypassing routes reaches the location on the intersection of its Bezier-curve used when the hole was encountered and the boundary of that hole (∂-*hole*), the information about the delay incurred in each route is reported back to the node that detected the hole first.
(iii) The information is used to
    (a) select the direction of bypassing (i.e. the one with a smaller extra-delay) and
    (b) notify (by back-propagation) the source that some of the Bezier curves from the family can no longer assure that the expected routing delay will be $\leq \Delta$. The source will stop using the stream-pipe that violates Δ.

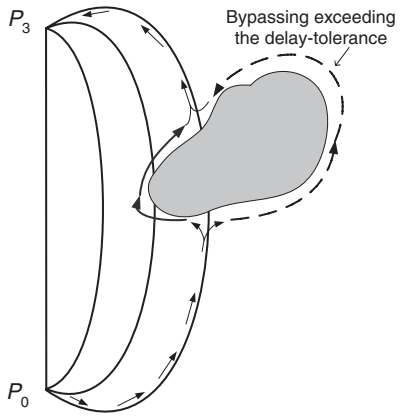


FIGURE 14. Bypassing holes.

An illustration of this problem is provided in Fig. 14. As shown, the counter-clockwise bypassing of the hole incurs an unacceptable delay overhead.

6. EXPERIMENTAL EVALUATION

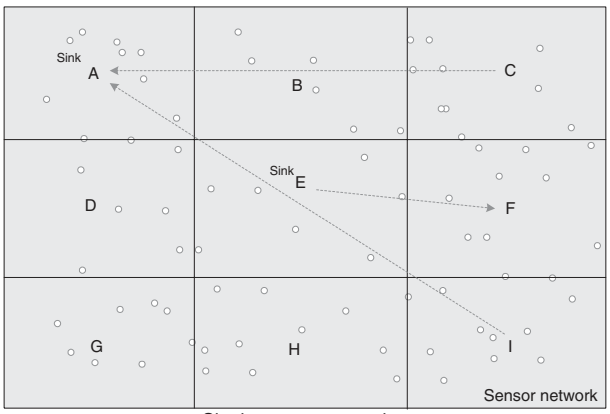
In order to demonstrate the advantages of our proposed methodology, we compared it with three routing techniques:

(i) 'k-shortest path' (k-short) approach [14];
(ii) 'Electrostatic-Field-Routing' (EFR) approach [42];
(iii) 'Energy Conserving Grid Routing Protocol' (ECGRID) [43];

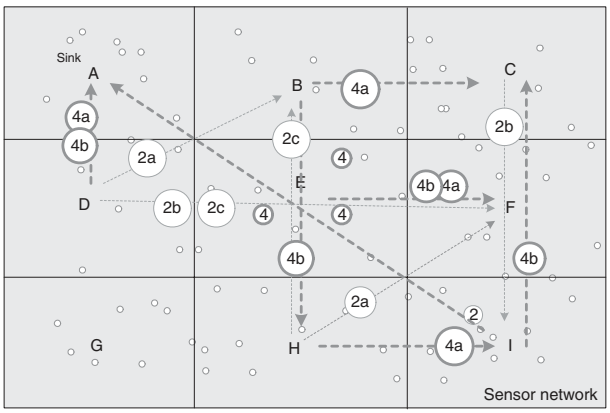
The first two techniques, k-short and EFR, are both similar in spirit with our approach, in the sense that they try to achieve lifetime extension through workload balancing among a collection of distinct paths. Each of them, however, has certain limitations when compared with the Bezier-based multi-path routing:—the k-short approach does not focus on the workload balancing around the sink and source node areas;—the EFR approach does not bound the admissible lengths of the routes, which can cause unacceptable delay in the transmission since, in the extreme case, the routes can span the networks' physical boundaries. Moreover, EFR also suffers from path-merging effects, thereby reducing the degree of workload balancing, whereas Bezier routing, with its stream-pipe design and node-forwarding policy, avoids path-merging whenever the network status allows it.

The last approach that we considered, ECGRID [43] is, in a sense, orthogonal to the 'mainstream' multi-path approaches. Its design actually focuses on ensuring that a communication path is maintained in an *ad hoc* network, by relieving the nodes whose energy reserves have dropped below a certain threshold. Essentially, the ECGRID partitions the space of interest into cells (hence 'grid'), and varies the selection of the node(s) within a given cell that are in charge of a particular routing path. As such, ECGRID bears a resemblance of the variations that our approach performs within a stream-pipe corresponding to one particular Bezier curve, however, it does not have the concept of a collection of alternating routes.

The experiments were performed using SIDnet-SWANS [44, 45], which is built upon the JiST-SWANS [46] simulator, adapted for WSN settings. The core objective of these experiments was to analyze the net effect of sustained routing over the lifetime of the network, taking in to consideration the nodes' death rate due to battery depletion, as well as the implications on the QoS in terms of packet delay. Experiments were based on a vast configuration space, which takes into consideration the following parameters: (i) sampling rate; (ii) network coverage; (iii) number of simultaneous queries and (iv) alternation policies. We configured three single-source scenarios, based on different placements of the source; and six multi-source scenarios (three with two-sources and three with four-sources), which are illustrated in Fig. 15. A summary of the parameters is given in Table 1, based on which nearly 4000



Single-source scenarios:
C->A, I->A, F->E



Multi-source scenarios :
(2) (a) D->B, H -> F;(b)D->F, C->I; (c)D->F, H->B
(4) (a)D->A,B->C,E->F,H->I
(b)D->A,B->H,E->F,I->C
(c) G->C,A->I,D->F,H->B

FIGURE 15. Deployments utilized for (a) single-source and (b) multiple-source scenarios.

TABLE 1. Summary of experimental configuration space.

No. queries	Query overlap (%)	Sampling interval (ms)	Alternation policy	No. packets sent before route change
1	0	4000	RANDOM	1
2	50	2000	SWEEP	10
4	80	1000	CT-MID-DIST	100
		500		

experiments have been automatically generated and performed. We note that the queueing capacity of each node was set to 120 packets, and we observed that the sampling rate of 500 m leads to queueing much more often than the other values.

The experimental platform consisted of a cluster of four quad core Linux machines, on which the experiments ran in a continuous batch. The testbed consists of 500 nodes randomly distributed using a uniform distribution function in an area of $6000 \times 6000\ ft^2$. Nodes are homogeneous, sharing the same configuration: 20 000 bps transmission/reception rate on Mac802.15.4; 10 s time-to-sleep interval; and power consumption characteristics based on the Mica2 Motes, as summarized in Table 2. A small battery powers each node, with an initial capacity of 35 mAh, which, given the power consumption characteristics from Table 2, is expected to power a node for a few tens of hours, depending on the load. The smaller battery size was chosen in order to reduce the simulation time while still preserving the validity of the experimental observations.

Figure 16 illustrates the amount of sensor nodes that stop functioning over time due to energy depletion, under a sustained sampling and packet transmission rate at the source nodes. As it can be observed, on the average, the Bezier-based approach has a gain of 5 h (approximately 20%) over k-short. This gain can be observed in Fig. 16 by focusing on the maximum time interval in which both algorithms yield the same number of energy-depleted nodes. Another way to observe the gains of the Bezier-based routing is by looking for the difference in energy-depleted nodes at any given instant of time. As shown, after 15 h, the number of dead nodes for Bezier-based routing is 30, whereas the k-short approach has almost 60 dead nodes and EFR has 80 dead nodes.

TABLE 2. Energy characteristics of *Mica2 Mote* (*MPR500CA*).

State	Based on (mA)	Energy requirement (mJ/ms)
Sensing active	$I_s = 10$	0.03
Sensing passive	$I_s = 0$	0
CPU active	$I_p = 8$	0.024
CPU idling	$I_i = 0.015$	$4.5 * 10^{-5}$
RADIO transmitting	$I_t = 27$	0.081
RADIO receiving	$I_r = 10$	0.03
RADIO listening	$I_l = 3$	0.009
RADIO Off-Mode	$I_{slp} = 0.5$	0.0015

An important observation is in order regarding Fig. 16. Namely, the bounds on the benefits of Bezier-based routing in terms of the number/rate of dead nodes needs to be considered in a slightly wider context. As a specific example, note that when comparing with k-short based routing, it may appear that the benefits of the proposed approach are temporary and that k-short may achieve better performance towards the end of the simulation period. However, there is a complementary aspect that needs to be considered—Fig. 17 illustrates the *degradation* of the QoS-level at run-time, expressed as the percentage of data packets that have been received at the sinks. As one can observe, Bezier-based routing is dropping much fewer packets compared with the other approaches. The main implications of this is as follows. Every packet that is being dropped represents a potential energy saving for the nodes further along a given route, since those nodes will not spend any energy for subsequent routing of that packet. Hence, in Fig. 16, some of the energy savings achieved by the other approaches are actually due to not performing the same effective transmission as the Bezier-based one.

To get an intuitive idea what would have been the benefits of the Bezier-based routing assuming that the packet-drop rate is the same, one can apply a linear scaling, factoring in the (relative weight of the) dropped packets in each approach. The resulting graph is illustrated in Fig. 18.

Both Figs. 16 and 18 show that ECGRID is the worst performing approach, and the main reason for it is that it does not exactly perform alternation among multiple routes for the purpose of load balancing over time. Although throughout the lifetime of the network different routes may be generated, each subsequent route becomes active only after the nodes servicing the previous route have reached a certain pre-defined low threshold in their battery reserves. As the number of available nodes decreases, ECGRID starts changing routes more often, which incurs a large overhead since many control messages need to be exchanged upon switching from one route to another. This is why ECGRID spends significantly more energy toward the end of the simulation time. Specifically, Fig. 18 shows that under ECGRID routing, the nodes start dying at a nearly exponential rate toward the end of the simulation time—for instance, after 20 h of simulation time, the ECGRID has depleted approximately 2.8 times as many nodes as Bezier-based routing.

Another important aspect that we investigated through our experiments is the benefits brought by Bezier-based routing in terms of lifetime extension. As we have mentioned (cf. [4, 5]), the definition of the lifetime depends on the particular application. Figure 19 illustrates the lifetime gain according to

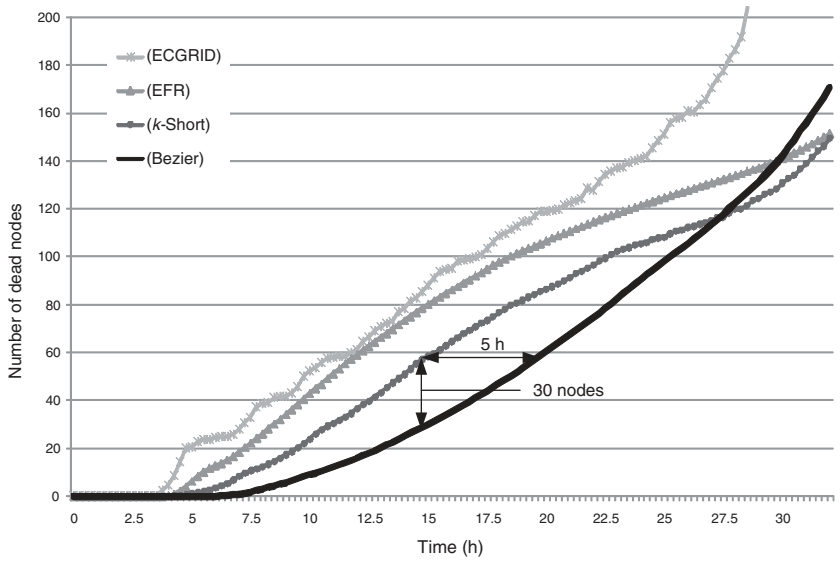


FIGURE 16. Number of dead nodes (averaged rate).

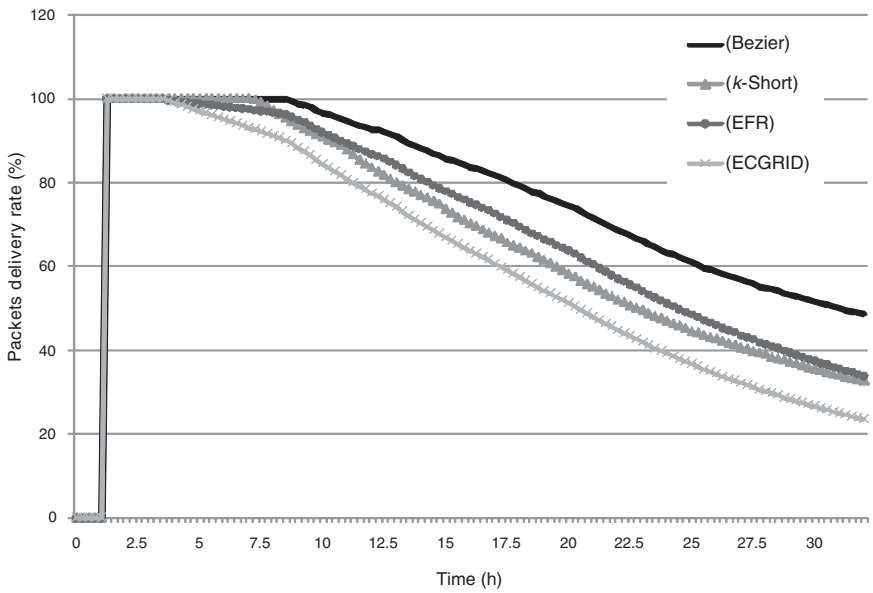


FIGURE 17. Successful packet transmission rate.

three definitions, based on the time until: (i) the first node dies, (ii) the first 1% of the nodes die and (iii) the first 10% of the nodes die. One can observe that the Bezier approach yields an improvement of between 20% and 34% over *k*-short, its best competitor.

We also analyzed the effects of the different approaches when the network-wide perceived degradation of performance in terms of packet drop rates was used as a QoS criterion. We considered three thresholds in packet drop rates: 1% (for sensitive application), 10% and 50%, and for each of them we present the duration of the acceptable operational regimes in Fig. 20. As shown, the Bezier-based approach gains between 20% and 42% of lifetime over *k*-short, which is, once again, the second-best method.

An important observation is that defining the lifetime of a WSN solely based on the *amount* of dead nodes may not be the most plausible approach. Namely, it is not only *how many* nodes are effectively dead, but also what their *distribution* is like with respect to the network. Hence, we looked at the improvement of load balancing by evaluating the standard deviation of the *residual energy* of the relay nodes within the network. Figure 21 shows that, at a peak-to-peak comparison, Bezier-based routing

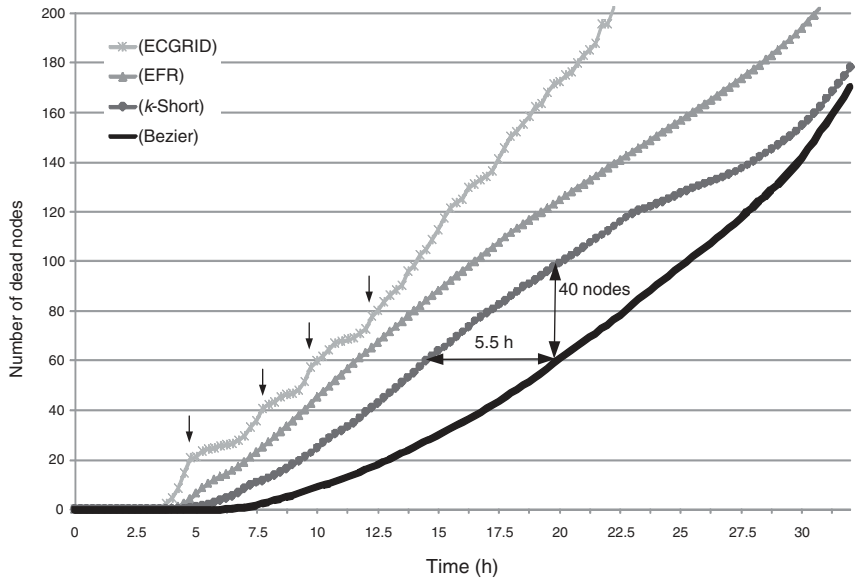


FIGURE 18. Scaled rate of dead nodes (factoring in the dropped packets).

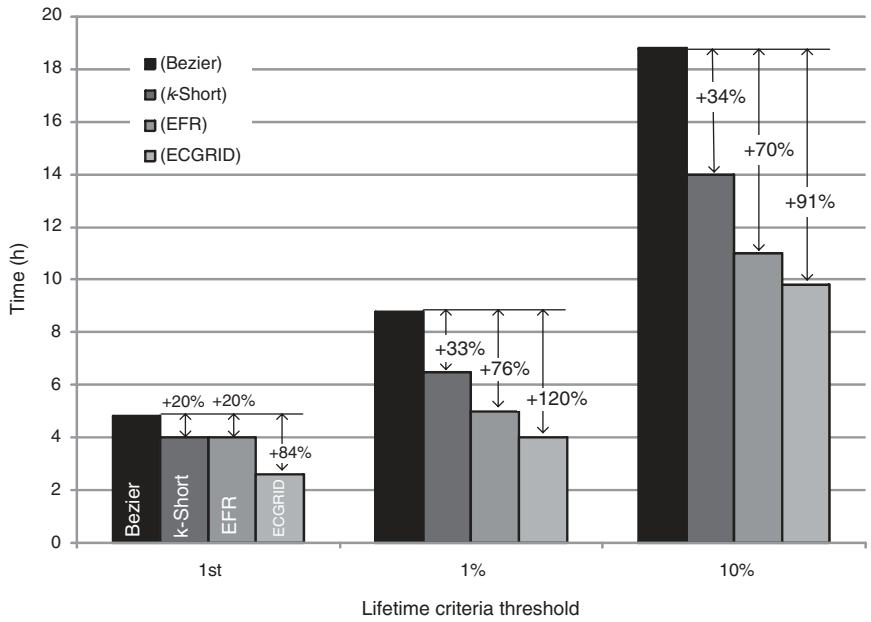


FIGURE 19. Lifetime performance according to dead nodes' count criteria.

achieves significant improvements over ECGRID (104%), EFR (43%) and *k*-short (20%) in all scenarios. Toward the end of simulation time, the difference among the standard deviations is reduced because, at that time, the relay nodes reach a predefined low-battery threshold (5% of initial battery capacity), and disengage from further relaying duties. As an additional observation, the time at which the standard deviation reaches its (worst) peak is delayed in Bezier routing by 7 h, compared with ECGRID, and 2 h, when compared with *k*-short.

Another QoS aspect that may be of interest for the users is the latency the delivery of the packets. As we mentioned, the benefits in terms of extended lifetime come at the cost of having a family of routes that are longer than the shortest path between a given *(sink, source)* pair, and Fig. 22 illustrates the averaged values of the maximal packet delivery latencies for the approaches considered. As can be seen, both Bezier-based and *k*-short routings exhibit lower delivery latencies, nearly 40% smaller than the ones exhibited by EFR, and twice as much

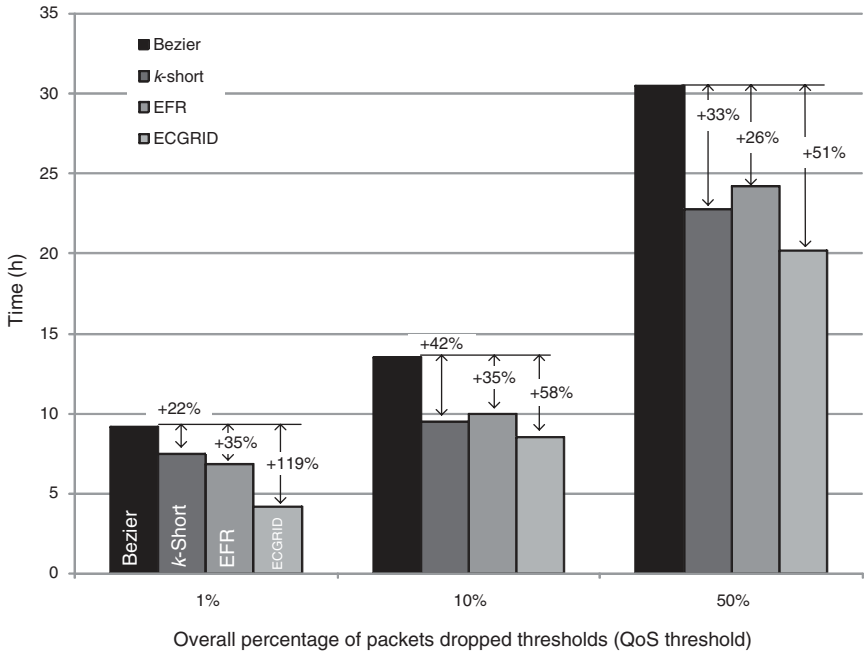


FIGURE 20. Lifetime and packets drop rate (QoS).

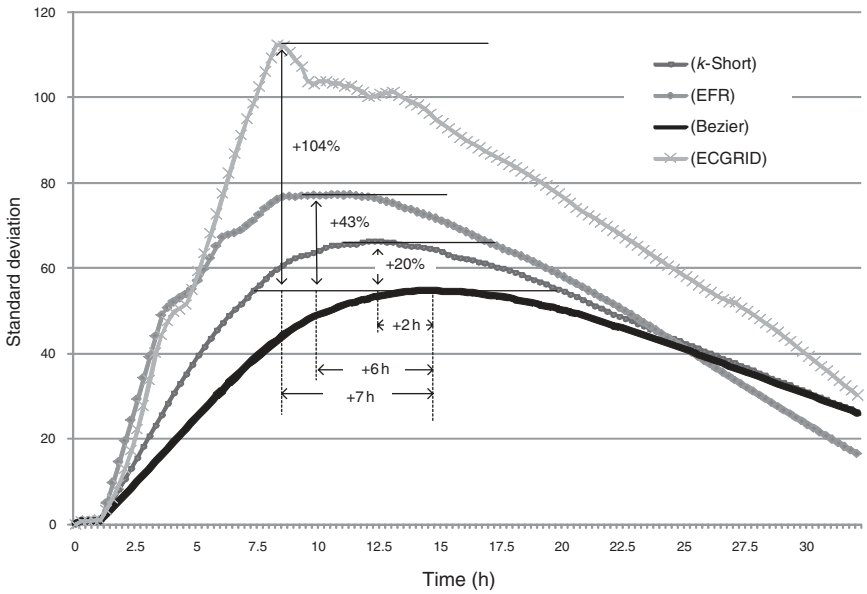


FIGURE 21. Standard deviation of energy levels as a measure of energy imbalance manifested in the entire network.

of ECGRID, throughout the time in which most of the nodes in the network are still operational. EFR routes, based solely on the gradient of the electrostatic field are not sensitive to the distance between the source and the sink. The Bezier-based and k-short routing exhibit comparable latency performance, since the control points of the outermost Bezier curve have been calibrated to roughly match the 'longest-admissible' path used by the k-short. EFR, however, exhibits the longest latencies since the protocol does not provide a mechanism to explicitly bound the spread of the alternating routes, and a significant number of routes will tend to spread throughout the entire network deployment area. An interesting observation is that toward the end of the simulation time, at which point the network has drastically changed in terms of the available nodes that can be used for routing, the latencies for both Bezier-based and k-short approaches tend to increase, whereas they

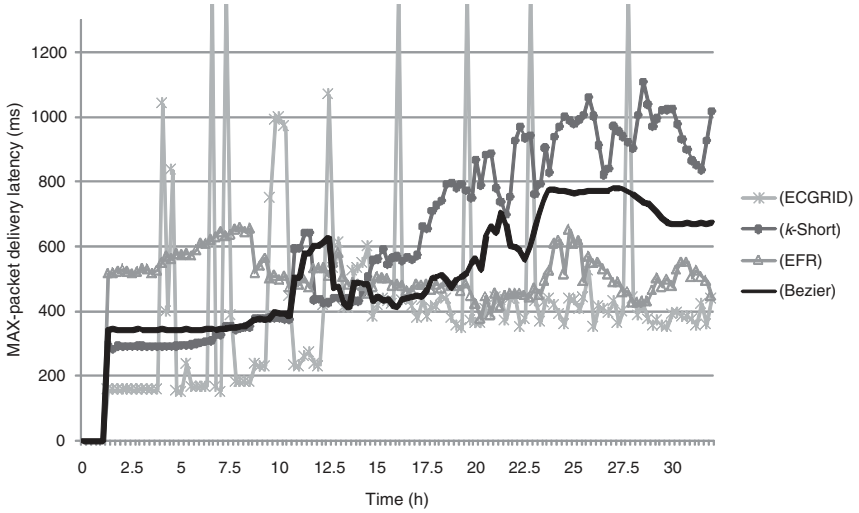


FIGURE 22. Packet delivery latencies.

decrease for the EFR. This is due to the tendency of EFR to initially use the boundary nodes and progressively build shorter routes as the outer nodes die, gradually shifting the route locations toward the center, thereby reducing the latency. Another interesting observation is the behavior of the ECGRID approach: its latency fluctuates very frequently, yielding many peaks and valleys in the graph in Fig. 22. The mean reason for this phenomenon is that, when switching routes, ECGRID relies on unacknowledged broadcast messages for the purpose of re-electing grid heads, which are not guaranteed to succeed immediately. Hence, the data-packets of the given query may need to be temporarily queued. Once a new route has been successfully established, the latency drops since there is no alternation among routes until the battery reserves of the nodes on the current one are exhausted. We note that some of the peaks of high latency were in the order of minutes.

We conclude this section with an observation regarding the benefits of the Bezier-based routing in the vicinity of the source (resp. sink) nodes. Namely, for all practical purposes, when the nodes that are within one hop from the source (resp. sink) die, the particular query pertaining to that given (*sink, source*) pair can no longer be executed. As we indicated throughout the paper, one of the advantages if the Bezier curves is that they enable exploiting a larger portion of the neighbors in the area around the vicinity of the source (resp. sink). In this specific experiment, to illustrate the benefits of such flexibility, we have fixed the boundary of the outermost routes for the k-short to be segments with a total length twice the size of the shortest path between the source and the sink, generating the angle of $2 * 60^\circ = 120^\circ$ as a boundary of the nodes around the sink and the source. We compared this against the Bezier curves with the same delay of the outermost curve; however, we allowed the angle bounding the one-hop neighbors of the source and

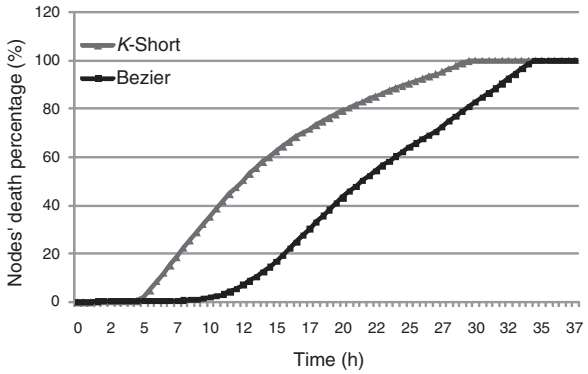


FIGURE 23. Percentage of dead one-hop neighbors.

the sink to be 240°, using the locations of the source and the sink as the final location of the respective control points (cf. Equation (10) in Section 4). As illustrated in Fig. 23, the death rate, in terms of percentage of the depleted nodes, among the one-hop neighbors of the source (resp. sink), after a certain initial period (5 h), is much higher for the k-short approach. In addition, the available nodes expire sooner, rendering the entire multi-path collection unusable for routing.

7. RELATED WORK

Many different aspects of routing protocols in WSN settings have been subject of extensive recent research work, and a survey encompassing different topics is presented in [3]. A complementary body of work was dedicated to the various issues related to the lifetime of WSNs, which have recently been surveyed in [4]

The concept of using multiple paths for routing purposes in sensor networks is not new. Results based on the Directed Diffusion approach [47], which considered multiple paths are presented in [14]. In addition to constructing disjoint paths that do not intersect, the work also considers *braided* paths which may intersect. Although some of the goals are similar to ours (prolonging lifetime), the work is more focused on resilience to failures and it proposes routings for which the decisions are made locally. Our work is, in a sense, complementary because although we consider multiple routes, we also incorporate the alternating among a family of routes and the load balancing in the vicinity of the source/sink node.

A different approach for extending the lifetime of a sensor network, which we used for our experiments in Section 6, is presented the [43]. Again, the key idea is to achieve a more energy-efficient routing by partitioning the region of interest into a grid, and keep one node per region awake for monitoring the existing activities and incoming queries, whereas switching off the radios of all the other nodes and putting them in a sleep state. Our approach also allows the nodes to power off their radios, but selectively keeps active only the nodes that are involved in an ongoing query. This work has been compared with [43], however, other approaches that implement multi-path routing exist, such as the one presented in [48]. The latter is not concerned with lifetime issues and does not consider the bounded delivery delay as a QoS indicator, and its main focus is on the overhead of route construction and maintenance, as well as the probability of error-packets due to link failures.

Our proposed approach is in a same category with the ones that use the concept of a TBF [23] and TBF+ [33]. These works focus on designing routing protocols where the nodes locally decide about forwarding of the packets, based on a pre-specified trajectory. In a sense, Niculescu and Nath [23] discusses a richer set of scenarios than are considered here (e.g. discovery of topology; broadcast; multi-cast), whereas we focus on a simpler setting of a long-running continuous query for given (*sink,source*) pairs. Further generalization is provided in [33], which introduces the TBF+ concept. TBF+ uses polynomial curves for the trajectories (e.g. an ellipse). Using curve fitting and the information about the energy of a particular node, TBF+ forces the trajectories to pass through points with higher energy reserves. Our Bezier-based approach uses algebraic parameterized curves and develops a formalism that enables constructing a family of such curves for the purpose of generating alternate routes. Furthermore, it offers the possibility of linking the alternative routes with the QoS criteria of acceptable temporal delay while prolonging the lifetime [5] of the network. It would be interesting to combine our solution with the ideas in [23] for the purpose of taking the current energy of the nodes into consideration when constructing a particular alternate route, which is the goal of our ongoing work.

The concept of virtual channels within a stream-pipe that we used, is somewhat similar to the concept of meta-paths introduced in [36]. In that work, a strip of size 0.86 times the communication radius is used, in order to balance the load among the nodes along a given route, with guaranteed bounds on the length of the longest route and the maximum load, whereas in our work, we considered the width of the stream-pipes as yet another parameter in the proposed methodology. The aspects of energy and power awareness for routing purposes have been considered in the literature from a couple of perspectives [15, 18, 49–51]. Balancing the energy awareness with the time constraints in a purely local manner was considered in [50], which introduces the priority-based stateless routing as an implementation of the volunteer-forwarding paradigm. A more general overview of the performance optimization via multi-path routing is presented in [15]. When it comes to database-like TinyDB queries, it has been demonstrated that the problem of finding the most energy-efficient routing structure is NP-complete [49]. Although we did not consider the problem of load balancing for multiple queries/requests, our work perceived the load balancing from two perspectives: across multiple routes, and within a single stream-pipe for a given *(sink, source)* pair. However, the parameterization of the Bezier curves makes our approach attractive from the perspective of local and distributed computations, as well as extending the number of nodes within the neighborhood of the sink and the source. We observe that some recent results presented in [51], interestingly, demonstrate that the impact of the particular routing algorithm on the energy consumption is larger in MANETs than in WSN settings. A work that dynamically combines tree-based structures with multi-path routing is presented in [16], however, the role of the multi-path portion is to increase the robustness (in terms of packets delivery), which is complementary to the goals of this work.

The problem of optimizing the sleeping schedule of the sensor nodes has been addressed in [38]. In particular, the work considers a detection of so-called rare event where the delay of event detection (due to sleep scheduling) is balanced with the lifetime extension of the network. Additionally, the problem of dynamic wake-up protocols which ensure satisfaction of spatio-temporal constraints when data is to be delivered in mobile environments has been considered in [52]. Our work is complementary to this line of works, in the sense that we have considered settings in which a family of routing curves is used (spatial extension) for the purpose of extending the lifetime of a WSN, while providing guarantees on the end-to-end communication delay of individual packets (temporal bound).

Many recent works have introduced various geometric concepts targeted toward solving problems of interest for routing and data management in sensor networks. For example, Bruck *et al.* [53] proposed distributed algorithms for constructing the medial axis of the sensor field which can subsequently be used as a 'reference map' for orientation and for localized algorithms for route construction. Subsequently, Wang *et al.* [54] extended the usage of the medial axis for the purpose of detecting the boundary nodes of a given sensor field. In complementary works [55, 56], the concepts like Yao and

Gabriel graphs were used for the purpose of developing power-efficient distributed algorithms for maintaining the connectivity of a given sensor network. In some sense, our work can be viewed as similar in spirit because we are using a geometric concept, namely, the Bezier curve. However, our goal is orthogonal since we focused on the problem of lifetime extension via alternate-path routing.

8. CONCLUDING REMARKS

We presented an approach for multi-path routing between a given (*sink,source*) pair, based on Bezier curves and geared toward load balancing among the relay nodes, for the purpose of prolonging the sensor networks lifetime. Due to the flexibility of constructing their shapes, one important benefit of using these curves for route' construction is that they allow for a better exploitation of the nodes in the vicinities of the source and sink. A feature of the Bezier curves that is especially appealing is that by carefully selecting the control points, one can impact the bounds on the delay, relative to the shortest path between the source and the sink. Most importantly, the construction of the routes based on Bezier curves can be done locally, that is, with a small overhead in terms of parameters transmitted, each relay node can decide who should be the next hop along the particular route toward the sink. To increase the balance of the distribution of the load among the nodes along a particular route, we introduced the concept of the stream-pipes and virtual channels.

We have experimentally compared the benefits of the Bezier-based routing methodology with three different approaches: *k*-short [14], EFR [42] and ECGRID [43], using our publicly available [44] SIDnet-SWANS simulator. The experiments, using several different parameters, have demonstrated that the proposed routing methodology indeed prolongs the lifetime of the network, while maintaining balanced energy levels of the nodes employed. Contrary to the observations for *ad hoc* networks [13], we have shown that non-negligible energy savings and extensions of lifetime can be achieved by judiciously controlled multi-path routing. As an implicit observation, the experiments for the ECGRID protocol provided a quantitative illustration of the known fact that the pure energy-oriented savings are not equivalent with the network's lifetime. Namely, the nodes' death rate of the ECGRID becomes extremely high after a certain time interval. In addition to providing an environment for evaluations of the analytical results in different experimental settings, one of the benefits of this work was the development of the different features of our simulator [45].

9. FUTURE WORK

There are several immediate extensions of this work. Currently, we are investigating the problem of the adaptability of the family of curves representing the alternate routes when multiple queries are simultaneously present within the geographic region of interest [57], for the purpose of reducing the energy expenses while maintaining the routing structures. As a first step, we are addressing the problem of reducing the number of intersecting paths due to multiple queries and its impact on the packet drop rate. In this context, we are also considering the case in which the sink requests measurements from a given geographical region. In such settings, it is very likely that some combination of trees and multi-paths will be needed, and we are extending the results in [58] for the purpose of prolonging the network lifetime via selection of such combinations. Another aspect of our ongoing work is investigating the problem of more dynamic adjustment of the selection of the nodes within a particular stream-pipe and analyzing its impact on the energy savings[2] in a detailed manner. In addition, we are planning to address issues related to location uncertainty—specifically, how it affects the determination of the corresponding stream-pipes.

From the end-user perspective, an interesting question is how the approach proposed in this paper could be incorporated into some formal algebra for specifying the desired properties of the routing (cf. [37, 49])? An important extension, both as a feature to our simulator as well as a more general tool for constructing paths that will avoid known holes, is to incorporate the existing results on 'reverse-engineering' of the Bezier (and other categories of splines) curves—given a particular shape, construct a curve which approximates it well with a minimum number of control points [20], where an interesting variant is the problem of bypassing holes with a minimum length of the route(s). We are also working on developing extensions of our SIDnet-SWANS simulator that will perform the following:

(i) Simulate the behavior of the underlying real network without actually interfering with its operations. This can be used for various hypothetical queries and scenarios that would involve 'what-if' type of reasoning about the impact of particular activities (e.g. new requests, physical removal of nodes) before placing the actual burden on the underlying network.
(ii) Efficiently (balancing the time and energy spent) obtain the information about the *state* of an underlying sensor network by querying a minimal subset of the nodes, whenever there is an indication that the simulation model has a large discrepancy with respect to the actual state of the nodes/network [60].

FUNDING

This research has been supported by the NSF grant CNS-0910952.

[2] Haenngi [59] presents 18 reasons why long hops may be preferred to small ones.

REFERENCES

[1] Akyildiz, I., Su, W., Sankarasubramaniam, Y. and Cayirci, E. (2002) Wireless sensor networks: a survey. *Comput. Netw.*, **38**, 393–422.

[2] F. Zhao, L.G. (2004) *Wireless Sensor Networks: An Information Processing Approach*. Elsevier/Morgan-Kaufmann.

[3] Akkaya, K. and Younis, M.F. (2005) A survey on routing protocols for wireless sensor networks. *Ad Hoc Netw.*, **3**, 325–349.

[4] Dietrich, I. and Dressler, F. (2009) On the lifetime of wireless sensor networks. *ACM Tans. Sensor. Netw.*, **5**, 1–39.

[5] Dong, Q. (2005) Maximizing System Lifetime in Wireless Sensor Networks. *IPSN*, Los Angeles, CA, USA, pp. 13–19.

[6] Kalpakis, K., Dasgupta, K. and Namjoshi, P. (2003) Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks. *Comput. Netw.*, **42**, 697–716.

[7] Kadayif, I. and Kandemir, M.T. (2004) Tuning In-Sensor Data Filtering to Reduce Energy Consumption in Wireless Sensor Networks. *DATE*, Paris, France, pp. 852–857.

[8] Lin, S., Gunopulos, D., Kalogeraki, V. and Lonardi, S. (2005) A Data Compression Technique for Sensor Networks with Dynamic Bandwidth Allocation. *TIME*, pp. 186–188.

[9] Shiou, C.-W., Lin, F.Y.-S., Cheng, H.-C. and Wen, Y.-F. (2005) Optimal Energy-Efficient Routing for Wireless Sensor Networks. *AINA*, Taipei, Taiwan, pp. 325–330.

[10] Park, J. and Sahni, S. (2006) Maximum lifetime routing in wireless sensor networks. *IEEE Trans. Comput.*, **55**, 609–619.

[11] Sharaf, M.A., Labrinidis, A. and Chrysanthis, P.K. (2008) Scheduling continuous queries in data stream management systems. *PVLDB*, Auckland, New Zealand, Vol. 1.

[12] Sharma, D., Chryaanthis, P. and Zadorozhny V. (2005) Timely Data Delivery in sn using Whirlpool. *DMSN Workshop*, Trondheim, Norway.

[13] Ganjali, Y. and Keshavarzian, A. (2004) Load Balancing in Ad Hoc Networks: Single-Path Routing vs. Multi-path Routing. *INFOCOM*, Hong Kong, China.

[14] Ganesan, D., Govindan, R., Shenker, S. and Estrin, D. (2001) Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *Mob. Comput. Commun. Rev.*, **5**, 11–25.

[15] Luo, W., Liu, W. and Zhang, Y. (2005) Performance optimization using multipath routing in mobile ad hoc and wireless sensor networks. *Comb. Optim. Commun. Netw.*, **2**, 117–146.

[16] Manjhi, A., Nath, S. and Gibbons, P.B. (2005) Tributaries and Deltas: Efficient and Robust Aggregation in Sensor Network Streams. *SIGMOD Conf.*, Baltimore, MD, USA, pp. 287–298.

[17] Parissidis, G., Lenders, V., May, M. and Plattner, B. (2006) Multi-path Routing Protocols in Wireless Mobile Ad Hoc Networks: A Quantitative Comparison. *NEW2AN*, St. Petersburg, Russia, pp. 313–326.

[18] Wu, S. and Candan, K.S. (2007) Power-aware single- and multipath geographic routing in sensor networks. *Ad Hoc Netw.*, **5**, 974–997.

[19] Thulasiraman, P., Ramasubramanian, S. and Krunz, M. (2007) Disjoint Multipath Routing to Two Distinct Drains in a Multi-drain Sensor Network. *INFOCOM*, Anchorage, AK, USA, pp. 643–651.

[20] Farin, G.E. (1993) Tighter convex hulls for rational bézier curves. *Comput. Aided Geom. Des.*, **10**, 123–125.

[21] van Dam, A. Fiener, S., Hughes, J and Foley, J.D. (1995) *Computer Graphics: Principles and Practice*. Addison-Wesley.

[22] Trajcevski, G., Ghica, O. and Scheuermann, P. (2006) Car: Controlled Adjustment of Routes and Sensor Networks Lifetime. *MDM*, Nara, Japan.

[23] Niculescu, D. and Nath, B. (2003) Trajectory based Forwarding and its Applications. *MOBICOM*, San Diego, CA, USA, 260.

[24] Nagpal, R., Shrobe, H.E. and Bachrach, J. (2003) Organizing a Global Coordinate System from Local Information on an Ad Hoc Sensor Network. *IPSN*, Palo Alto, CA, USA, pp. 333–348.

[25] Fang, L., Du, W. and Ning, P. (2005) A Beacon-less Location Discovery Scheme for Wireless Sensor Networks. *INFOCOM*, Miami, FL, USA, pp. 161–171.

[26] Reichenbach, F., Born, A., Timmermann, D. and Bill, R. (2006) A Distributed Linear Least Squares Method for Precise Localization with Low Complexity in Wireless Sensor Networks. *DCOSS*, San Francisco, USA.

[27] Bulusu, N., Estrin, D., Girod, L. and Heidemann, J. (2001) Scalable Coordination for Wireless Sensor Networks: Self-configuring Localization Systems. *Proc. 6th Int. Symp. Communication Theory and Applications* (*ISCTA*), Toronto, Ontario, Canada, July.

[28] Srinivasan, V., Nuggehalli, P., Chiasserini, C.-F. and Rao, R.R. (2003) Cooperation in Wireless Ad Hoc Networks. *INFOCOM*, Ambleside, Lake District, UK.

[29] Higham, N. (1998) *Handbook of Writing for the Mathematical Sciences*. SIAM.

[30] Farin, G. (1990) *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press.

[31] de Casteljau, P. (1959) *Curbes a Poles*. INPI.

[32] Alt, H., Welzl, E. and Wolfers, B. (1997) Piecewise Linear Approximation of Bézier-Curves. *Symp. Computational Geometry*, Nice, France, pp. 433–435.

[33] do Val Machado, M., Goussevskaia, O., Mini, R.A.F., Rezende, C.G., Loureiro, A.A.F., Mateus, G.R. and Nogueira, J.M.S. (2005) Data Dissemination using the Energy Map. *WONS*, St. Moritz, Switzerland, pp. 139–148.

[34] de Berg, M., van Kreveld, M., Overmars, M. and Schwarzkopf, O. (2008) *Computational Geometry—Algorithms and Applications* 2nd ed. Springer.

[35] Fang, Q., Gao, J. and Guibas, L.J. (2004) Locating and Bypassing Routing Holes in Sensor Networks. *INFOCOM*, Hong Kong, China.

[36] Gao, J. and Zhang, L. (2006) Load-balanced short-path routing in wireless networks. *IEEE Trans. Parallel Distrib. Syst.*, **17**, 377–388.

[37] Zadorozhny, V., Chrysanthis, P.K. and Labrinidis, A. (2004) Algebraic Optimization of Data Delivery Patterns in Mobile Sensor Networks. *DEXA Workshops*, Zaragoza, Spain, p. 668.

[38] Cao, Q., Abdelzaher, T.F., He, T. and Stankovic, J.A. (2005) Towards Optimal Sleep Scheduling in Sensor Networks for Rare-Event Detection. *IPSN*, UCLA, Los Angeles, CA, USA, pp. 20–27.

[39] Andrews, M. and Zhang, L. (2004) Routing and Scheduling in Multihop Wireless Networks with Time-Varying Channels. *SODA*, New Orleans, LA, USA, pp. 1031–1040.

[40] Chiasserini, C.-F. and Rao, R.R. (2001) Combining Paging with Dynamic Power Management. *INFOCOM*, Anchorage, AK, USA, pp. 996–1004.

[41] Karp, B. and Kung, H.T. (2000) Gpsr: Greedy Perimeter Stateless Routing for Wireless Networks. *MobiCom'00: Proc. 6th Annual Int. Conf. Mobile Computing And Networking*, New York, NY, USA, pp. 243–254. ACM Press.

[42] Kalantari, M. and Shayman, M. (2004) Routing in Wireless Ad Hoc Networks by Analogy to Electrostatic Theory. *IEEE Int. Conf. Communications*, Paris, France, Vol. 7, pp. 4028–4033.

[43] Chih-Min Chao, C.-T.H. and Sheu, J.-P. (2003) Energy-conserving grid routing protocol in mobile Ad Hoc networks. *32nd Int. Conf. Parallel Processing (ICPP 2003)*, 6–9 October 2003, Kaohsiung, Taiwan, pp. 399–413.

[44] http://www.ece.northwestern.edu/peters/sensors/.

[45] Ghica, O., Trajcevski, G., Scheuermann, P., Bischoff, Z. and Valtchanov, N. (2008) Sidnet-swans: A Simulator and Integrated Development Platform for Sensor Networks Applications. *SenSys.* Proc. 6th Int. Conf. Embedded Networked Sensor Systems, SenSys 2008, Raleigh, NC, USA, November 5–7, 2008, pp. 385–386.

[46] http://jist.ece.cornell.edu/index.html.

[47] Intanagonwiwat, C., Govindan, R. and Estrin, D. (2000) Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. *MOBICOM*, Boston, MA, USA, pp. 56–67.

[48] Pham, P. (2003) Performance Analysis of Reactive Shortest Single-Path and Multi-Path Routing Mechanism with Load Balance. *INFOCOM*, San Franciso, CA, USA.

[49] Buragohain, C., Agrawal, D. and Suri, S. (2005) Power Aware Routing for Sensor Databases. *INFOCOM*, Miami, FL, USA.

[50] Xu, Y., Lee, W.-C. and Xu, J. Energy-aware and time-critical geo-routing in wireless sensor networks. *Int. J. Distrib. Sensor Netw.* (*IJDSN*), **4**, 317–346.

[51] Yoon, S., Dutta, R. and Sichitiu, M.L. (2007) Power Aware Routing Algorithms for Wireless Sensor Networks. *ICWMC'07: Proc. 3rd Int. Conf. Wireless and Mobile Communications*, p. 15, Washington, DC, USA. IEEE Computer Society.

[52] Bhattacharya, S., Xing, G., Lu, C., Roman, G.-C., Chipara, O. and Harris, B. (2005) Dynamic Wake-up and Topology Maintenance Protocols with Spatiotemporal Guarantees. *IPSN*, Los Angeles, CA, USA, pp. 28–34.

[53] Bruck, J., Gao, J. and Jiang, A. (2005) Map: Medial Axis Based Geometric Routing in Sensor Networks. *MOBICOM*, Cologne, Germany, pp. 88–102.

[54] Wang, Y., Gao, J. and Mitchell, J.S.B. (2006) Boundary Recognition in Sensor Networks by Topological Methods. *MOBICOM*, Los Angeles, CA, USA, pp. 122–133.

[55] Li, X.-Y., Wan, P.-J. and Frieder, O. (2003) Coverage in wireless ad hoc sensor networks. *IEEE Trans. Comput.*, **52**, 753–763.

[56] Song, W.-Z., Wang, Y., Li, X.-Y. and Frieder, O. (2005) Localized algorithms for energy efficient topology in wireless ad hoc networks. *MONET*, **10**, 911–923.

[57] Chipara, O., Lu, C. and Stankovic, J. (2006) Dynamic conflict-free query scheduling for wireless sensor networks. *IEEE Int. Conf. Network Protocols* (*ICNP*), Santa Barbara, CA, USA, pp. 321–331.

[58] Trajcevski, G., Ghica, O., Scheuermann, P., Tamassia, R. and Cruz, I.F. (2008) Alternating Multiple Tributaries + Deltas. *DMSN*, Auckland, New Zealand, August 24, 2008, pp. 28–34.

[59] M. Haenngi, D.P. (Oct. 2005) Routing in ad hoc networks: A case for long hops. *IEEE Commun. Mag.*, **43**, 93–101.

[60] Girod, L., Elson, J., Cerpa, A., Stathopoulos, T., Ramanathan, N. and Estrin, D. (2004) Emstar: A Software Environment for Developing and Deploying Wireless Sensor Networks. *USENIX Annual Technical Conf., General Track*, Boston, MA, USA, pp. 283–296.