SinkTrail: A Proactive Data Reporting Protocol for Wireless Sensor Networks

Xinxin Liu, *Student Member*, *IEEE*, Han Zhao, *Student Member*, *IEEE*, Xin Yang, and Xiaolin Li, *Member*, *IEEE*

Abstract—In large-scale Wireless Sensor Networks (WSNs), leveraging data sinks' mobility for data gathering has drawn substantial interests in recent years. Current researches either focus on planning a mobile sink's moving trajectory in advance to achieve optimized network performance, or target at collecting a small portion of sensed data in the network. In many application scenarios, however, a mobile sink cannot move freely in the deployed area. Therefore, the precalculated trajectories may not be applicable. To avoid constant sink location update traffics when a sink's future locations cannot be scheduled in advance, we propose two energy-efficient proactive data reporting protocols, SinkTrail and SinkTrail-S, for mobile sink-based data collection. The proposed protocols feature low-complexity and reduced control overheads. Two unique aspects distinguish our approach from previous ones: 1) we allow sufficient flexibility in the movement of mobile sinks to dynamically adapt to various terrestrial changes; and 2) without requirements of GPS devices or predefined landmarks, SinkTrail establishes a logical coordinate system for routing and forwarding data packets, making it suitable for diverse application scenarios. We systematically analyze the impact of several design factors in the proposed algorithms. Both theoretical analysis and simulation results demonstrate that the proposed algorithms reduce control overheads and yield satisfactory performance in finding shorter routing paths.

Index Terms—Wireless sensor networks, mobile sink, data gathering, routing, logical coordinates

1 Introduction

Wireless Sensor Networks (WSNs) have enabled a wide spectrum of applications through networked low-cost low-power sensor nodes, e.g., habitat monitoring [15], precision agriculture [11], and forest fire detection [27]. In these applications, the sensor network will operate under few human interventions either because of the hostile environment or high management complexity for manual maintenance. Since sensor nodes have limited battery life, energy saving is of paramount importance in the design of sensor network protocols.

Recent research on data collection reveals that, rather than reporting data through long, multihop, and error-prone routes to a static sink using tree or cluster network structure, allowing and leveraging sink mobility is more promising for energy efficient data gathering [1]. Mobile sinks, such as animals or vehicles equipped with radio devices, are sent into a field and communicate directly with sensor nodes, resulting in shorter data transmission paths and reduced energy consumption.

However, data gathering using mobile sinks introduces new challenges to sensor network applications. To better benefit from the sink's mobility, many research efforts have been focused on studying or scheduling movement patterns of a mobile sink to visit some special places in a deployed area, in order to minimize data gathering time. In such approaches a mobile sink moves to predetermined sojourn points and query each sensor node individually. Although several Mobile Elements Scheduling (MES) protocols have been proposed to achieve efficient data collection via controlled sink mobility [14], [21], [28], determining an optimal moving trajectory for a mobile sink is itself an NP-hard problem [14], and may not be able to adapt to constrained access areas and changing field situations. Take the precision agriculture application as an example, as shown in Fig. 1, where mobile sinks collecting data mainly follow trails or field boundaries in order not to damage crops, and change trajectories dynamically according to farmland situations.

Typically, without scheduling the trajectory for a mobile sink in advance, a data gathering protocol using mobile sinks suggests that a mobile sink announce its location information frequently throughout the network. Many Sink-Oriented Data Dissemination (SODD) protocols use such approach, e.g., Directed Diffusion [8], Declarative Routing Protocol (DRP) [3], and GRAB [25], whereas different aggregation methods may be adopted. This class of methods is referred to as SODD in the literature [24]. This approach is much more flexible in terms of sinks' movement, but incurs significant control message overheads. An example data reporting path of SODD is presented using black solid route shown in Fig. 2. In addition to large amount of energy consumption on flooding control messages, change of routing paths due to the sinks' movement, and energy cost on detouring large data packets (originally targeted at the previous sink location, now changed to the current sink location) severely impair protocol performances. This problem can be alleviated by transmitting data via the shortest route to the mobile sink's future

- *X. Liu, H. Zhao, and X. Yang are with the Computer and Information Science and Engineering Department, University of Florida, 406 New Engineering Building, Gainesville, Florida 32611-6200. E-mail: {xinxin, han, xin}@cise.ufl.edu.*
- *X. Li is with the Department of Electrical and Computer Engineering, University of Florida, 216 Larsen Hall, PO Box 116200, Gainesville, Florida 32611-6200. E-mail: andyli@ece.ufl.edu.*

Manuscript received 24 July 2011; revised 23 Sept. 2011; accepted 28 Sept. 2011; published online 18 Oct. 2011.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2011-06-0423.
Digital Object Identifier no. 10.1109/TC.2011.207.

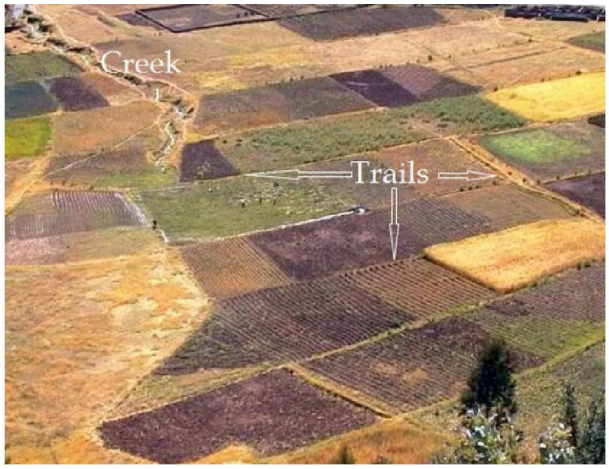


Fig. 1. A photograph showing a typical farmland of irregular shapes. A mobile sink's movement in such environment is constrained. By courtesy of http://www.hat.net/.

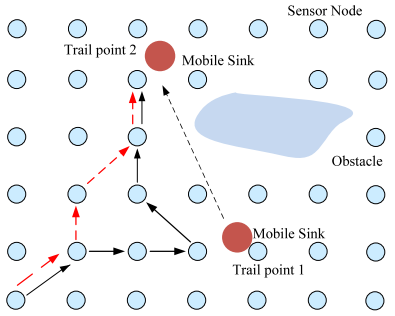


Fig. 2. Comparison of routing methods: SODD versus SinkTrail. A mobile sink moves from the trail Point 1 to the trail Point 2 in a sensor network deployed in a wild area. The black solid route represents the data reporting path in SODD (without location prediction); it takes six hops to reach the trail Point 2. Whereas red dashed route represents predictive SinkTrail routing, taking only four hops to reach the mobile sink's current location.

locations, as observed in the red dashed route in Fig. 2. Therefore, if sensors can predict the mobile sink's movement, the energy consumption would be greatly reduced and data packets handoff would be smoother.

In this paper, we propose SinkTrail, a proactive data reporting protocol that is self-adaptive to various application scenarios, and its improved version, SinkTrail-S, with further control message suppression. In SinkTrail, mobile sinks move continuously in the field in relatively low speed, and gather data on the fly. Control messages are broadcasted at certain points in much lower frequency than ordinarily required in existing data gathering protocols. These sojourn positions are viewed as "*footprints*" of a mobile sink. Considering each footprint as a virtual landmark, a sensor node can conveniently identify its hop count distances to these landmarks. These hop count distances combined represent the sensor node's coordinate in the logical coordinate space constructed by the mobile sink. Similarly, the coordinate of the mobile sink is its hop count distances from the current location to previous virtual landmarks. Having the destination coordinate and its own coordinate, each sensor node greedily selects next hop with the shortest logical distance to the mobile sink. As a result, SinkTrail solves the problem of movement prediction for data gathering with mobile sinks.

Our contributions in this paper are manifold.

1. We propose a unique logical coordinate representation for tracking mobile sinks without assistance of GPS devices or predefined landmarks, which is widely applicable to various network settings and scenarios.
2. We design a novel low-complexity dynamic routing protocol for data gathering with one or multiple mobile sink(s), which effectively reduces average route length and cuts down total energy consumption.
3. We propose and evaluate an enhanced SinkTrail protocol, called SinkTrail-S.
4. We conduct extensive comparison studies and simulations with popular existing solutions.

The rest of the paper is organized as follows. Section 2 presents related work. Detailed protocol design is introduced in Section 3. Section 4 presents analytical and simulation results, and demonstrates the advantages of SinkTrail algorithms over previous approaches. The impact of several design factors of SinkTrail is investigated and analyzed in Section 5. Section 6 concludes the paper.

2 Related Work

Leveraging data sinks' mobility in sensor data collection has been a topic of tremendous practical interests and drawn intensive research efforts in the past few years. The most challenging part of this approach is to effectively handle the control overheads introduced by a sink's movement. At the first look, broadcasting a mobile sink's current location to the whole network is the most natural solution to track a moving mobile sink. This type of approach is sink oriented and some early research efforts, e.g., [3], [8], [25], have demonstrated its effectiveness in collecting a small amount of data from the network. Several mechanisms have been suggested to reduce control messages. The TTDD protocol, proposed in [24], constructed a two-tier data dissemination structure in advance to enable fast data forwarding. In [7], a spatial-temporal multicast protocol is proposed to establish a delivery zone ahead of mobile sink's arrival. Control messages are flooded to wake up nodes in the delivery zone. Similarly, Park et al. [17] proposed DRMOS that divides sensors into "wake-up" zones to save energy. Fodor and Vidács [5] lowered communication overheads by proposing a restricted flooding method; routes are updated only when topology changes. Luo and Hubaux [13] proposed that a mobile sink should move following a circle trail in deployed sensor field to maximize data gathering efficiency. One big problem of the multicasting methods lies in its flooding nature. Moreover, these papers either assume that mobile sinks move at a fixed velocity and fixed direction, or follow a fixed moving pattern, which largely confines their application. The SinkTrail protocol with message suppression minimizes the flooding effect of control messages without confining a mobile sink's movement, thus is more attractive in real-world deployment. Another solution utilizes opportunistic data reporting. For instance, in [19], Shah and Shakkottai studied data collection performance when a mobile sink presents at random places in the network. The method relies heavily on network topology and density, and suffers scalability issues when all data packets need to be forwarded in the network.

Another category of methods, called Mobile Element Scheduling (MES) algorithms [4], [14], [21], [22], [23], [28], [29], [30], considered controlled mobile sink mobility and advanced planning of mobile sink's moving path. Ma and Yang [14] focused on minimizing the length of each data gathering tour by intentionally controlling the mobile sink's movement to query every sensor node in the network. When data sampling rates in the network are heterogeneous, scheduling mobile sinks to visit hot-spots of the sensor network becomes helpful. Example algorithms can be found in [4], [22], and [23]. Although the MES methods effectively reduce data transmission costs, they require a mobile sink to cover every node in the sensor field, which makes it hard to accommodate to large scale and introduces high latency in data gathering. Even worse, finding an optimal data gathering tour in general is itself an NP-hard problem [12], [14], and constrained access areas or obstacles in the deployed field pose more complexity. Unlike MES algorithms, SinkTrail, with almost no constraint on the moving trajectory of mobile sinks, achieves much more flexibility to adapt to dynamically changing field situations while still maintains low communication overheads.

SinkTrail uses sink location prediction and selects data reporting routes in a greedy manner. In [9], Keally et al. used sequential Monte Carlo theory to predict sink locations to enhance data reporting. SinkTrail employs a different prediction technique that has much lower complexity. Moreover, SinkTrail does not rely on the assumption of location-aware sensor nodes, which could be impractical for some real-world applications. The routing protocol of SinkTrail is inspired by recent research on virtual coordinate routing [2], [6], [16], [18]. Rao et al. [18] proposed a greedy algorithm for data reporting using logical coordinates rather than geographic coordinates. Fonseca et al. [6] presented vector form virtual coordinates, in which each element in the vector represented the hop count to a landmark node. SinkTrail adopts this vector representation and uses past locations of the mobile sink as virtual landmarks. To the best of our knowledge, we are the first to associate a mobile sink's "footprints" left at moving path with routing algorithm construction. The vector form coordinates, called trail references, are used to guide data reporting without knowledge of the physical locations and velocity of the mobile sink.

3 SinkTrail Protocol Design

3.1 Problem Formulation

We consider a large scale, uniformly distributed sensor network $\mathbb{N}$ deployed in an outdoor area. Fig. 3 shows an example deployment. Nodes in the network communicate with each other via radio links. We assume the whole sensor network is connected, which is achieved by deploying sensors densely. We also assume sensor nodes are awake when data gathering process starts (by synchronized schedule or a short "wake up" message). In order to gather data from $\mathbb{N}$, we periodically send out a number of mobile sinks into the field. These mobile sinks, such as robots or vehicles with laptops installed, have radios and processors to communication with sensor nodes and processing sensed data. Since energy supply of mobile sinks can be replaced or recharged easily, they are assumed to have unlimited power. A data gathering process starts from the time mobile sinks enter the field and terminates when: either 1) enough data are collected (measured by a user defined threshold); or 2) there are no more data report in a certain period. The SinkTrail protocol is proposed for sensor nodes to proactively report their data back to one of the mobile sinks. To illustrate our data gathering algorithm clearly, we first consider the scenario where there is only one mobile sink in $\mathbb{N}$. The multiple mobile sinks scenario is discussed in Section 3.3.

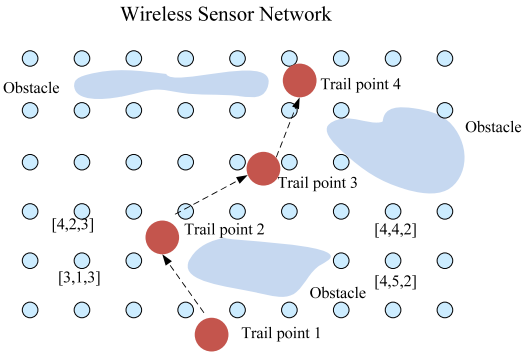


Fig. 3. Data gathering with one mobile sink: large solid dots indicate the mobile sink's trail points, and sensor nodes maintain trail references as logical coordinates. Shaded areas stand for obstacles.

3.2 SinkTrail Protocol with One Mobile Sink

During the data gathering process, the mobile sink moves around in $\mathbb{N}$ with, relatively, low speed, and keeps listening for data report packets. It stops at some places for a very short time, broadcasts a message to the whole network, and moves on to another place. We call these places *"Trail Points,"* and these messages *"Trail Messages."* Example trail points are shown in Fig. 3. Let $\overline{\tau}$ be the average transmission range. Apparently two adjacent trail points should be separated by a distance longer than $\overline{\tau}$, otherwise, the hop count information will not be significantly different. To facilitate the tracking of a mobile sink, we assume that the distances between any two consecutive trail points are same (or similar), denoted as $K\overline{\tau}, K \geq 1$. However, distribution of these trail points does not necessarily follow any pattern. A trail message from a mobile sink contains a sequence number (*msg.seqN*) and a hop count (*msg.hopC*) to the sink. The time interval between a mobile sink stops at one trail point and arrives at the next trail point is called one *"move."* There are multiple moves during a data gathering round. The tasks of a mobile sink is summarized in Algorithm 1.

Algorithm 1. Mobile sink's strategy

```
/*——Initialization——*/
msg.seqN ← 0;
msg.hopC ← 0;
Announces step size parameter K
/*——Moving strategies——*/
while Not get enough data or Not timeout do
    Move to next trail point;
    msg.seqN← msg.seqN +1;
    Stop for a very short time to broadcast trail message;
    Concurrently listen for data report packets;
end while
End data gathering process and exit;
```

TABLE 1
Notations for the SinkTrail Protocol

n_i	sensor node i
N	total number of sensor nodes in $\mathbb{N}$
S	mobile sink
M	number of mobile sinks
v_i	trail reference of node i
${e_i}^j$	the jth element in v_i
d_v	trail reference size, $d_v = \|\|v\|\|$
b	average number of neighbors of each node
λ	the most recent message sequence number a node has recorded
π_i	the ith trail point of S
Π	the collection of trail points
D_π	total number of trail points
K	step size parameter for one move (a step of K hop counts is $K\bar{r}$)
T_i	timer duration of node i

In the SinkTrail algorithm, we use vectors called *"Trail References"* to represent logical coordinates in a network. The trail reference maintained by each node is used as a location indicator for packet forwarding. All trail references are of the same size. Notations used throughout the protocol description are listed in Table 1.

The data reporting procedure consists mainly two phases. The first phase is called *logical coordinate space construction*. During this phase, sensor nodes update their trail references corresponding to the mobile sink's trail messages. After d_v hop counts have been collected, a sensor node enters the *greedy forwarding* phase, where it decide how to report data packets to the mobile sink.

3.2.1 Logical Coordinate Space Construction

At beginning, all sensor nodes' trail references are initialized to $[-1, -1, \ldots, -1]$ of size d_v. A special variable λ that is used to track the latest message sequence number is also set to -1. After the mobile sink S enters the field, it randomly select a place as its first trail point π_1, and broadcasts a trail message to all the sensor nodes in $\mathbb{N}$. The trail message, $<msg.seqN, msg.hopC>$, is set to $<1, 0>$, indicating that this is the first trail message from trail point one, and the hop count to S is zero.

The nodes nearest to S will be the first ones to hear this message. By comparing with λ, if this is a new message, then λ will be updated by the new sequence number. And node n_i's trail reference v_i is updated as follows: first, every element in v_i is shifted to left by one position. Then, the hop count in the received trail message is increased by one, and replaces the right-most element ${e_i}^{d_v}$ in v_i. After n_i updated its trail reference, this trail message is rebroadcasted with the same sequence number and an incremented hop count. The same procedure repeats at all the other nodes in $\mathbb{N}$. Within one move of S, all nodes in the network have updated their trail references according to their hop count distances to S's trail point π_1. If a node receives a trail message with a sequence number equals to λ, but has a smaller hop count than it has already recorded, then the last hop count field in its trail reference is updated, and this trail message is rebroadcasted with the same sequence number and an incremented hop count. Trail messages that has sequence number less than λ will be discarded to eliminate flooding messages in the network. The steps described in Algorithm 2 summarizes the operations to update a trail reference. During the data gathering procedure, a node's trail reference needs to be updated every time a new trail message is received.

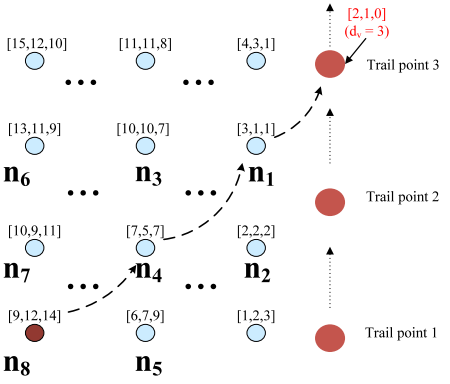


Fig. 4. Example execution snapshot of SinkTrail: large solid dots indicate trail points and its moving path.

Algorithm 2. Trail reference update algorithm

while *Data gathering process is not over* **do**
  /*——Receive a trail message——*/
  if msg.seqN $> \lambda$ **then**
    $\lambda \leftarrow$ msg.seqN;
    Shift v_i to left by one position;
    ${e_i}^{d_v} \leftarrow$ msg.hopC $+ 1$;
    msg.hopC $\leftarrow$ msg.hopC $+1$;
    Rebroadcast message;
  else if msg.seqN $= \lambda$ **then**
    Compare ${e_i}^{d_v}$ with (msg.hopC $+ 1$);
    if ${e_i}^{d_v} >$ (msg.hopC $+ 1$) **then**
      ${e_i}^{d_v} \leftarrow$ msg.hopC $+ 1$;
      msg.hopC $\leftarrow$ msg.hopC $+1$;
      Rebroadcast message;
    else
      Discard the message;
    end if
  else if msg.seqN $< \lambda$ **then**
    Discard the message;
  end if
end while
/*——Reset Variables——*/
For $j = 1, \ldots, d_v$ ${e_i}^j \leftarrow -1$;
$\lambda \leftarrow -1$

After each node in the network received d_v distinct trail messages, the logical coordinate space is established. A snapshot of a part of the network $\mathbb{N}$ is shown in Fig. 4. Trail references, such as $[3, 1, 1]$ or $[2, 2, 2]$, are considered logical coordinates of the sensor nodes in a network.

3.2.2 Destination Identification

SinkTrail facilitates the flexible and convenient construction of a logical coordinate space. Instead of scheduling a mobile sink's movement, it allows a mobile sink to spontaneously stop at convenient locations according to current field situations or desired moving paths. These sojourn places of a mobile sink, named trail points in SinkTrail, are footprints left by a mobile sink, and they provide valuable information for tracing the current location of a mobile sink.

Considering these footprints as virtual landmarks, hop count information reflects the moving trajectory of a mobile sink. A logical d_v-dimensional coordinate space is then established.

One advantage of SinkTrail is that the logical coordinate of a mobile sink keeps invariant at each trail point, given the continuous update of trail references. This is because the mobile sink's hop count distance to its previous $d_v - 1$ footprints are always $K(d_v - 1), K(d_v - 2), \ldots, K$, and 0 to its current location. Therefore, the logical coordinate $[K(d_v - 1), K(d_v - 2), \ldots, 0]$ represents the current logical location of the mobile sink. We call this coordinate "*Destination Reference*." This destination reference does not necessarily require a mobile sink to have linear moving trajectory. Although arbitrary movement of a mobile sink may deteriorate the accuracy of destination reference, it can still serve as a guideline for data reporting. Here, we set $K = 1$ and $d_v = 3$ to ease our presentation. A large value of K means even less broadcast frequency. The impacts of mobile sinks' moving pattern and broadcast frequency are investigated in Section 5. In Fig. 4, assume S is at the trail Point 3 now, then its destination reference should be $[2, 1, 0]$. When S moves to the trail Point 4, the coordinate space is updated based on trail Points 2, 3, and 4, and destination reference of the mobile sink is still $[2, 1, 0]$.

3.2.3 Greedy Forwarding

Once a node has updated the three elements in its trail reference (we use $d_v = 3$ for easy understanding and clear presentation), it starts a timer that is inverse proportional to the right-most element in its trail reference. For example, node n_5's trail reference is $[6, 7, 9]$ in Fig. 4, then the duration of its timer is set to $T_5 = T_{init} - \mu \times 9$. Here, T_{init} and μ are predefined constants. The choice of timer function, T_{init}, and μ may vary. However, we assume the timer durations are significantly longer than the propagation time of a trail message, so that timers on all nodes are viewed as starting at the same time. The timer mechanism is mainly used to differentiate data reporting orders (another usage is discussed in SinkTrail-S protocol); so the clock on each sensor node does not need to be perfectly synchronized. Since the right-most element in a node's trail reference is the latest hop count information from this node to a mobile sink, the inverse proportional timers ensure that nodes faraway from S have shorter timer durations than those close to S, thus will start data reporting first. When a node's timer expires, it initiates the data reporting process.

Every sensor node in the network maintains a routing table of size $O(b)$ consisting of all neighbors' trail references. This routing table is built up by exchanging trail references with neighbors, as described in Algorithm 3; and it is updated whenever the mobile sink arrives at a new trail point. Although trail references may not be global identifiers since route selection is conducted locally, they are good enough for the SinkTrail protocol. Because each trail reference has only three numbers, the size of exchange message is small. When a node has received all its neighbors' trail references, it calculates their distances to the destination reference, $[2, 1, 0]$, according to *2-norm* vector calculation, then greedily chooses the node with the smallest distance as next hop to relay data. If there is a tie the next hop node can be randomly selected. The complete procedure of greedy forwarding is presented in Algorithm 3. Take the network in Fig. 4 as an example, when node n_8 decides to report its data, it compares n_4, n_5, and n_7's vector distance with $[2, 1, 0]$. Since n_5 and n_7's distance to $[2, 1, 0]$ is $\sqrt{133}$ and $\sqrt{249}$, respectively, and n_4's distance is $\sqrt{90}$, n_4 is chosen as the next hop of n_8.

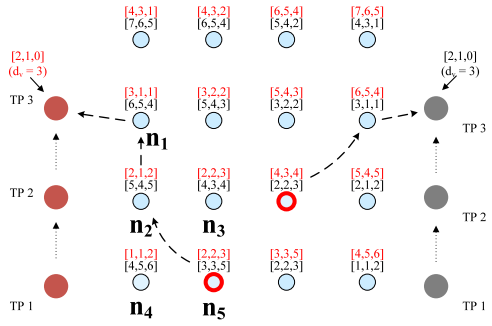


Fig. 5. Example execution snapshot of SinkTrail of multiple mobile sinks scenario.

Algorithm 3. Greedy data forwarding algorithm

```
/*——Start a timer——*/
if All elements of the trail reference are updated then
    Start timer T_i = T_init − μ × e_i^{d_v}
    Exchange trail references with neighbors
end if
/*——When timer expires——*/
Set destination as [(d_v − 1), ..., 2, 1, 0]
/*——Probe mobile sink——*/
if A mobile sink is within radio range then
    Send data to the mobile sink directly
else
    Choose the neighbor closest to destination as the next hop
    Forward all data to next hop
end if
```

3.3 SinkTrail Protocol with Multiple Mobile Sinks

The proposed SinkTrail protocol can be readily extended to multisink scenario with small modifications. When there is more than one sink in a network, each mobile sink broadcasts trail messages following Algorithm 1. Different from one sink scenario, a sender ID field, *msg.sID*, is added to each trail message to distinguish them from different senders.

Algorithms executed on the sensor node side should be modified to accommodate multisink scenario as well. Instead of using only one trail reference, a sensor node maintains multiple trail references that each corresponds to a different mobile sink at the same time. Fig. 5 shows an example of two mobile sinks. Two trail references, colored in black and red, coexist in the same sensor node. In this way, multiple logical coordinate spaces are constructed concurrently, one for each mobile sink. When a trail message arrives, a sensor node checks the mobile sink's ID in the message to determine if it is necessary to create a new trail reference. The procedure is summarized in Algorithm 4. In SinkTrail trail references of each node represent node locations in different logical coordinate spaces, when it comes to data forwarding, because reporting to any mobile sink is valid, the node can choose the neighbor closest to a mobile sink in any coordinate

space. Sink location in each logical coordinate space is still $[2, 1, 0]$, as we use $K = 1, d_v = 3$. If each mobile sink has a different K value, sensor nodes will calculate neighbors' distances to multiple destination references and select route accordingly. Detailed description is in Algorithm 5. It is well known that geographic routing and virtual coordinate-based routing ensure loop-free routes [6], [18], so does SinkTrail.

Algorithm 4. Trail reference update algorithm for multiple mobile sinks

while *Data gathering process is not over* **do**
  /*——Receive a trail message——*/
  if *New mobile sink ID* **then**
    Create v_i_mID;
    Create λ_mID;
  else
    /*——Message from a known sink——*/
    if msg.seqN $>$ λ_mID **then**
      Shift v_i*_mID to left by one position*;
      ${e_i}^{d_v} \leftarrow \text{msg.hopC} + 1$;
      msg.hopC $\leftarrow$ msg.hopC $+1$;
      Rebroadcast message;
    else if msg.seqN $=$ λ_mID **then**
      Compare ${e_i}^{d_v}$ *with* $(\text{msg.hopC} + 1)$;
      if ${e_i}^{d_v} > \text{msg.hopC} + 1$ **then**
        ${e_i}^{d_v} \leftarrow \text{msg.hopC} + 1$;
        msg.hopC $\leftarrow$ msg.hopC $+1$;
        Rebroadcast message;
      else
        Discard the message
      end if
    else if msg.seqN $<$ λ_mID **then**
      Discard the message;
    end if
  end if
end while
/*——Reset variables——*/
v_i_prototype $\leftarrow [-1, -1, \ldots, -1]$ of size d_v;
λ_prototype $\leftarrow -1$;

Algorithm 5. Greedy data forwarding algorithm for multiple mobile sinks

/*——Start a timer——*/
if *All elements of the trail reference are updated* **then**
  Start timer $T_i = T_{init} - \mu \times {e_i}^{d_v}$
  Exchange trail references with neighbors
end if
/*——When timer expires——*/
Set destination as $[(d_v - 1), \ldots, 2, 1, 0]$
/*——Probe mobile sink——*/
if *A mobile sink is within radio range* **then**
  Send data to the mobile sink directly
else
  Compare neighbors' trail references with destination reference in already established logical coordinates
  Choose the neighbor closest to any mobile sink as the next hop
  Forward all data to next hop
end if

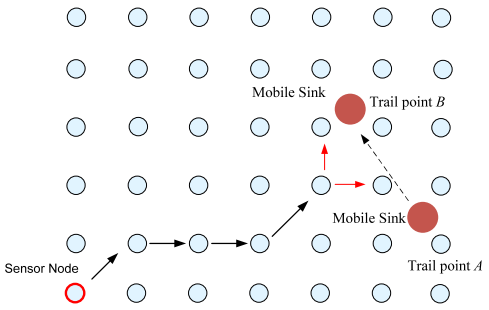


Fig. 6. An illustrative example to show that mobile sink's movement has less impact on remote sensor nodes than immediate ones.

Fig. 5 gives us an example of data gathering in multiple coordinate spaces. For node n_5, its neighbor node n_2's vector distance to $[2, 1, 0]$ with regard to the red mobile sink on the left is two, and $\sqrt{43}$ to the right gray mobile sink. And all other neighbors of n_5 has larger vector distance to the two sinks. So n_2 is used as the next hop to route to the red mobile sink.

3.4 SinkTrail-S Protocol

In SinkTrail, flooding trail messages to the whole network can be nontrivial in terms of energy consumption. To further optimize the energy usage and eliminate unnecessary control messages in the network, we propose SinkTrail-S algorithm as an improvement to the original SinkTrail. SinkTrail-S algorithm is mainly based on the following two observations. First, in a large-scale sensor network, the sensor nodes that are far away from a mobile sink may not be significantly affected by a single movement of the mobile sink. Take the sensor network shown in Fig. 6 as an example, when the mobile sink moves from trail point A to trail point B, the yellow sensor node at the left bottom corner may still have the same hop count distance to the mobile sink, and the routing path chosen from last "*move*" of the mobile sink may still be valid. In this case, the trail messages can be suppressed with high probability. Second, when a node has finished data reporting and forwarding, trail reference updating becomes meaningless and results in huge waste of energy, especially for peripheral sensor nodes. To properly handle these two situations, we propose a message suppression policy at a small cost of extra state storage at each sensor node. Each sensor node will compare the current hop count distance to a mobile sink with the most recently received one. If these two are same, it indicates the path length through the node to the mobile sink is still same, making it unnecessary to rebroadcast this trail message. In case of the second situation, each node maintains a state variable in its memory. When a node finishes data reporting, it marks itself as "*finished*," and informs all its neighbor nodes. A node stops trail reference updating and trail message rebroadcasting whenever itself and all its neighbors are "*finished*." Again, this method is guaranteed by the timer mechanism that ensures sequential data packets reporting order from network peripheral to a mobile sink's current location. For accidental situations due to timer failure, a new data packet may arrive at a node that has already stopped trail reference updating. In that case old trail references are used. This may cause a longer

routing path but the result is still acceptable for data reporting. Algorithm 6 presents a detail description of SinkTrail-S protocol.

Algorithm 6. Trail reference update with message suppression

while *Data gathering process is not over* **do**
  /*——Receive a trail message——*/
  if msg.seqN $> \lambda$ **then**
    $\lambda \leftarrow$ msg.seqN;
    if ${e_i}^{d_v} =$ msg.hopC $+ 1$ **then**
      Discard the message
    else
      Shift v_i to left by one position;
      ${e_i}^{d_v} \leftarrow$ msg.hopC $+ 1$;
      msg.hopC $\leftarrow$ msg.hopC $+1$;
      Rebroadcast message;
    end if
  else if msg.seqN $= \lambda$ **then**
    Compare ${e_i}^{d_v}$ *with* (msg.hopC $+ 1$);
    if ${e_i}^{d_v} >$ (msg.hopC $+ 1$) **then**
      ${e_i}^{d_v} \leftarrow$ msg.hopC $+ 1$;
      msg.hopC $\leftarrow$ msg.hopC $+1$;
      Rebroadcast message;
    else
      Discard the message;
    end if
  else if msg.seqN $< \lambda$ **then**
    Discard the message;
  end if
end while
/*——Reset Variables——*/
For $j = 1, \ldots, d_v$ ${e_i}^j \leftarrow -1$;
$\lambda \leftarrow -1$;

4 Performance Evaluation

There are many mobile sink oriented approaches for data collection in sensor networks, e.g., Directed Diffusion [8], TTDD [24], and GRAB [25]. These protocols, as in SinkTrail, do not pose any constraint on a mobile sink's movement, nor do they require any special setup phase, generaly referred to as sink-oriented data dissemination approaches. Although SODD approaches may apply different aggregation functions for better performance, similar strategies can be applied to SinkTrail as well. In order to gain more insights on the energy efficiency of SinkTrail, and to demonstrate the advantage of incorporating sink location tracking, we compare the overall energy consumption of SinkTrail with these protocols. Simulation results for SinkTrail-S are also presented to show further improved performance.

4.1 Theoretical Analysis

Before we proceed the following variables are defined for clear presentation and fair comparison. We consider a network $\mathbb{N}$ that consists of N sensor nodes and M mobile sinks. All the sensor nodes are data sources. We assume sensor nodes are deployed in a grid topology for ease of understanding. However, our analysis can be extended to other uniformly distributed topology. Therefore, the edge of the grid is roughly $\sqrt{N}$. Denote the energy cost for transmitting or receiving a control message be α,[1] and the cost for a data packet be β. We have $\beta >> \alpha$ since compared to trail messages, data packets are usually larger in terms of data size, which is proportional to the energy cost for radio transmission.

In SinkTrail, energy consumption mainly includes data packet forwarding cost, E_{data}, routing table maintenance cost, $E_{routing}$, and trail message transmission cost, E_{trail}.

Two factors affect the energy cost of data forwarding: number of data packets and the average route length. The number of data packets is determined by the number of data sources in a network, in this case, N. The average route length, on the other hand, may vary depending on the locations a mobile sink has traveled. We estimate an upper bound of the average route length by considering the situation that a mobile sink appears randomly at a location inside the deployed field. In this case, we can find $\frac{N}{2}$ pairs of sensor nodes that any one pair of nodes' distances to the mobile sink added up to at most $\sqrt{2N}$. Thus, the average route length should be upper bounded by $\frac{N}{2} \cdot \sqrt{2N}/N$. We use a coefficient c, where $0 < c \leq \frac{1}{2}$, to describe the average route length. Hence, we have

$$E_{data} = \beta \cdot c \cdot \sqrt{2N} \cdot N. \quad (1)$$

This energy cost upper bound for data reporting will not be affected by the number of mobile sinks, since every data reporting message will travel through the shortest possible path. Increased number of mobile sink will only decrease the total energy cost for data reporting.

According to SinkTrail protocol, the total number of trail messages depends on the network size, N, the number of trail points visited by each mobile sink, D_π, and the number of mobile sinks, M. The energy consumption for trail message transmission is given by

$$E_{trail} = \alpha \cdot M \cdot N \cdot D_\pi. \quad (2)$$

In SinkTrail, the energy consumption for each node to maintain local routing information is linearly proportional to the number of its neighbors, denoted by b. If there are multiple mobile sinks, the energy consumption increases as each node keeps a different trail reference for each mobile sink. Because of the broadcast nature of wireless media, this type of control message only needs to be transmitted once by each sensor node. Therefore, the energy cost for routing information maintenance is summarized by

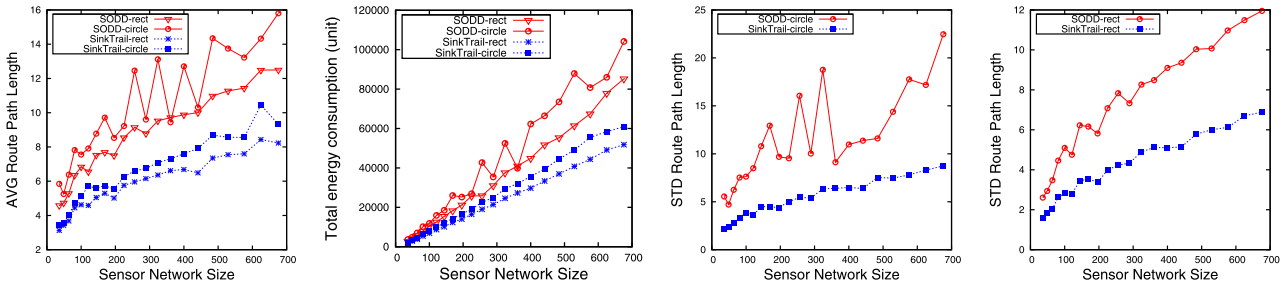
$$E_{routing} = \alpha \cdot N \cdot M. \quad (3)$$

The overall energy consumption of SinkTrail protocol is

$$E_{ST} = \beta \cdot c \cdot \sqrt{2N} \cdot N + \alpha \cdot M \cdot N \cdot D_\pi + \alpha \cdot N \cdot M. \quad (4)$$

TTDD [24] is a well-known data dissemination protocol that uses mobile sinks. Since in TTDD, data collection requests are initiated by mobile sinks, it in fact belongs to the category of SODD approach. The basic data gathering procedure of TTDD is preceded by a data grid setup phase,

1. In practice, energy cost for transmitting and receiving might be slightly different.



(a) Average route length (b) Energy consumption (c) STD of route length (d) STD of route length

Fig. 7. Performance comparison between SODD and SinkTrail. (a) Average routing path length. (b) Total energy consumption. (c) Route length variances: circular sink movement. (d) Route length variances: rectangular sink movement.

where each data source propagates some descriptive information about its data to the whole network in order to construct a grid structure for guiding the forward direction of query messages. When a mobile sink queries for certain data, this query message is only flooded within a grid cell, then the preconstructed data grid structure will help the query find the corresponding data source. According to this description, the total energy consumption of TTDD includes the following three components: energy used for grid constructing, E_{grid}; query flooding, E_{query}; and data reporting, E'_{data}.

The energy cost for data reporting in TTDD is determined by amount of data packets and length of routing paths. Since in TTDD, data packets are routed toward a mobile sink that appears randomly in the deployed field, the average route length is similar to SinkTrail. Therefore, we have

$$E'_{data} = \beta \cdot c \cdot \sqrt{2N} \cdot N. \quad (5)$$

According to TTDD protocol, the whole deployed area is divided into small cells. A query for data is only flooded inside one cell. However, as we are considering a data collection process that aims at getting all sensed data in the network, it is reasonable to argue that this single query will affect each of the data sources, thus will be propagated by all sensor nodes in the network. Therefore, we have

$$E_{query} = \alpha \cdot M \cdot N \cdot b_{cast}, \quad (6)$$

where b_{cast} is the number of such query broadcasts.

In the grid construction phase every data source in the network propagates a descriptive message about its data to the whole network, so that certain nodes will become anchors for a particular data source. Since every node is a potential data sources, the energy cost for this procedure is

$$E_{grid} = \alpha \cdot N \cdot N. \quad (7)$$

Adding the energy consumption of different tasks together, we have

$$E_{TTDD} = \beta \cdot c\sqrt{2N} \cdot N + \alpha \cdot M \cdot N \cdot b_{cast} + \alpha \cdot N \cdot N. \quad (8)$$

Comparing (4) and (8), we can see that one of the differences lies in the second term. Since D_π represents the number of broadcasts a mobile sink makes during the data gathering procedure in SinkTrail, and b_{cast} indicates the number of times a sink initiates a query in TTDD, these two variables can be set as equal. Thereafter, the difference can be ignored here. Another difference is between the routing information exchange cost and grid construction cost. Typically, the number of mobile sinks should be significantly less than total number of sensor nodes, i.e., $M << N$, and $\alpha \cdot N \cdot M << \alpha \cdot N \cdot N$. Hence, we have $E_{ST} < E_{TTDD}$, meaning that the total energy consumption of SinkTrail protocol is less than the energy cost by TTDD under the same condition.

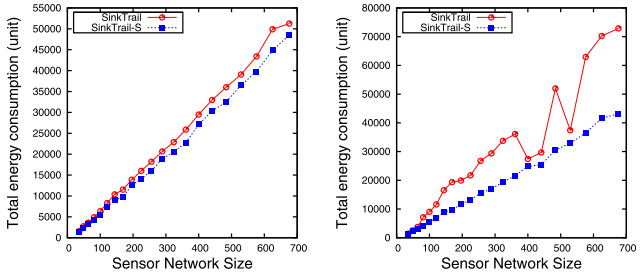
According to the theoretical analysis, we can see that TTDD's special grid setup phase facilitates a mobile sink to quickly collect a small amount of data in the network, unsuitable for collecting data from all sensors. The energy consumption of TTDD to collect all data includes energy consumption of the generalized SODD method plus a constant grid setup cost.

4.2 Simulation Results

We conducted extensive simulations for different scenarios using TOSSIM [10] to compare the performance of Sinktrail and SODD. SODD is implemented following the abstraction described in [24]. An extension to TOSSIM is implemented to allow us simulate the sink mobility following predefined moving patterns. Simulation of TTDD is omitted for the following reasons. First, TTDD falls into the category of general SODD approach, hence, generalized SODD approach can capture the main features of TTDD. Moreover, after the theoretical analysis, we can see that the grid setup phase in TTDD is targeted at querying specific data, and not very suitable to our data collection scenario.

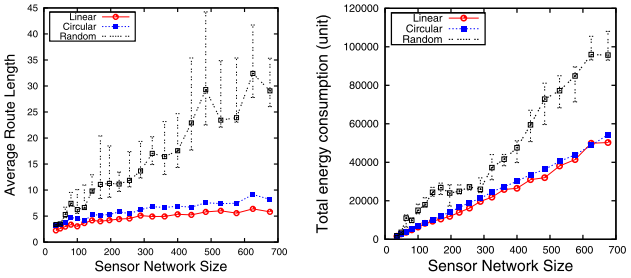
Note that SinkTrail-S is not included in this set of comparison. Another set of simulation results will be presented later to investigate the effectiveness of message suppression.

In the SODD approach, whenever a mobile sink moves to a different location, it broadcasts its current position to the whole network. As the message propagates a routing tree is established. Each node reports back its sensed data to parent node and finally, all data are merged at the root. This SODD approach suffers from losing track of the sink when location update is infrequent. To ensure fair comparison, a broadcast frequency higher than typically required by SinkTrail is used to ensure proper termination of SODD. We use one mobile sink in this set of simulations. The mobile sink moves in a rectangular or circular fashion in both algorithms. We set the data gathering threshold to 98 percent. From Figs. 7a and 7b, we observe that SinkTrail protocol



(a) Linear movement (b) Circular movement

Fig. 8. Performance comparison between SinkTrail protocol and SinkTrail-S protocol.



(a) Average route length (b) Energy consumption

Fig. 10. Impact of mobile sink's moving pattern.

outperforms SODD for every experimental network size. The energy consumption saving is on average 35.06 percent with a route length deduction of 33.80 percent for one sink SinkTrail. Figs. 7c and 7d show the standard deviation of the route lengths generated by SODD and SinkTrail protocols. As you can see, SinkTrail protocol results in a more even routing path length distribution throughout the network. All these results validate the conclusion that SinkTrail helps a mobile sink to achieve energy efficient data gathering in wireless sensor networks.

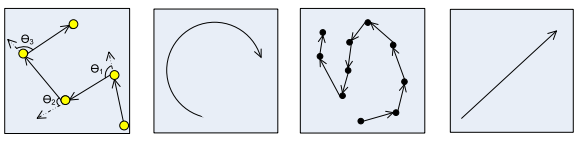
To demonstrate the effectiveness of message suppression in SinkTrail-S, we simulated SinkTrail-S with circular and linear sink moving patterns and compare the result with the basic SinkTrail protocol. It is worth noting that energy cost for informing neighbors is also counted in implementation. In Fig. 8, we observe that, although SinkTrail-S spends extra costs on state storage and informing message transmission, the method effectively reduces energy consumption in the investigated scenarios.

5 Impact Factors

5.1 Impact of Moving Patterns of a Mobile Sink

First, we examine how the moving pattern of a mobile sink can affect the energy consumption for data collection, as directional change in a mobile sink's movement is unavoidable due to occasional obstacles depicted in Fig. 3.

To numerically model the moves conducted by a mobile sink, we trace the moving trail of a mobile sink on a plain and measure the directional change at each trail point. Specifically, suppose at some time the mobile sink arrives at trail point $\pi_i \in \Pi$, we define the angular displacement θ_i as the angular variation of moving directions. Fig. 9a illustrates an example of recorded angular displacements at multiple trail points. As a result, the accumulative angular displacement of a mobile sink becomes a quantitative metric for the moving pattern. In Figs. 9b, 9c, and 9d,



(a) angular displacement (b) circular (c) random (d) diagonal

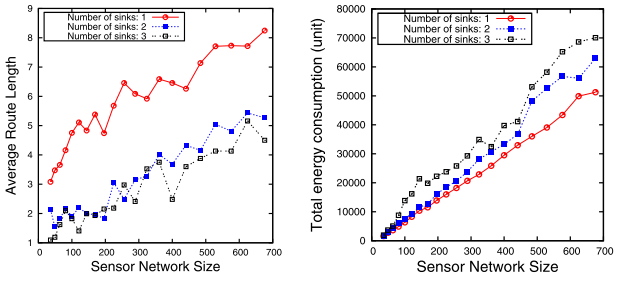
Fig. 9. Mobile sink moving pattern. (a) Angular displacement θ_i at each trail points. (b) Circular moving pattern, $\Sigma\theta_i$ is 360 degree. (c) Random moving pattern, $\Sigma\theta_i$ is greater than 360 degree. (d) Linear moving pattern, $\Sigma\theta_i$ is 0 degree.

we depict three representative moving patterns performed by a mobile sink. In simulation, we distributed sensor nodes in a grid topology. The network size varied from 6×6 sensor nodes to 26×26 (with step size 1). An empirical radio signal strength trace is loaded for each simulated sensor nodes. The radio sensitivity parameter is adjusted so that each node has about 5 to 12 neighbors, which is in accordance with realistic situations. We also designate β to be 20 times of α in the simulation [26]. The performance of SinkTrail is inspected in terms of average route length and overall energy consumption. Three moving patterns including *circular*, *random*, and *linear* moves are compared. The results are shown in Figs. 10a and 10b.

From these two figures, we observe that, both the average route length and energy consumption increase as the network size grows. For the three moving patterns, linear movement incurs the least energy consumption and the shortest average route length. As to the circular movement case, the mobile sink changes its direction regularly and smoothly, leading to performance close to the linear movement case. Finally, for the random move case, the results vary in a wide range that indicated by the dashed bars bounding the average values. This is because it is more difficult to track and predict the behavior of a randomly moving mobile sink. Therefore, SinkTail's overall performance may suffer greatly when the directional change is radical at some trail point. Although SinkTrail does not place any moving restriction in general, changing directions strategically in a smooth and regular manner is more beneficial than radical and unpredictable moving in SinkTrail.

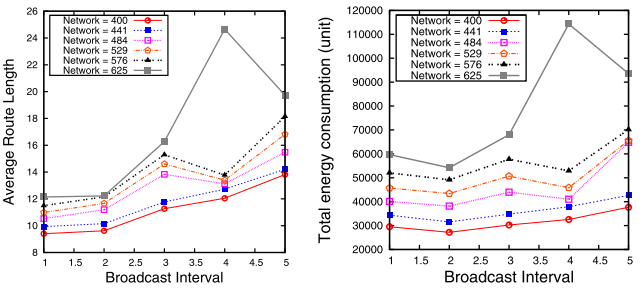
5.2 Impact of Number of Mobile Sinks

We are interested in finding out how the number of mobile sinks affects the overall system performance. In the scenario with multiple mobile sinks, several logical coordinate spaces are constructed concurrently and data packets are forwarded to the destination reference via the shortest path in any coordinate space. It is natural to think that increasing the number of mobile sinks reduces the average route length and thus reduces the total energy consumption. Nonetheless, more mobile sinks also impose heavier burdens for trail message broadcasting and routing information maintenance. Even worse, multiple number of mobile sinks in a network aggravate control traffic congestion and communication delays, which will in turn result in higher packet loss and retransmission rate. To acquire visualized results on the impact, we simulate the multiple mobile sinks



(a) Average route length (b) Energy consumption

Fig. 11. Impact of the number of mobile sinks.



(a) Average route length (b) Energy consumption

Fig. 12. Impact of broadcast frequency.

scenario using the aforementioned simulation setup. The number of mobile sinks used is up to three and they are injected into the network at the same time. For fair comparison all the mobile sinks moved randomly via different routes, and broadcasted at the same frequency. We averaged the results of 20 simulation runs and the results are exhibited in Figs. 11a and 11b. The trends shown in the figures confirm our analysis. The average route length is reduced by 46.54 and 53.70 percent for two and three sinks, respectively; while for the total energy cost, using more mobile sinks increases trail messages and routing table costs, thereby yield to 17.6 and 33.06 percent energy consumption increment for two and three sinks, respectively. Overall, defining route length deduction over extra energy cost as performance price ratio, we have 2.64 for two sinks and 1.62 for three sinks scenario. According to this, we conclude that adding multiple sinks is more suitable for applications with tight data gathering deadlines.

5.3 Impact of Broadcasting Frequency

The impact of sink broadcast frequency is two sided. If the mobile sink broadcasts its trail messages more frequently, sensor nodes will get more up-to-date trail references, which is helpful for locating the mobile sink. On the other hand, frequent trail message broadcast results in heavier transmission overheads. Suppose the time duration between two consecutive message broadcasting is Δt, we derive a general range of Δt to guide the proper implementation of SinkTrail and SinkTrail-S.

Assume the trail message is transmitted instantaneously, then Δt is determined by mobile sink's traveling time Δt_m between two consecutive trailing points and sojourn time Δt_s at each trail point

$$\Delta t = \Delta t_m + \Delta t_s. \tag{9}$$

Given the average mobile sink moving speed $\overline{\nu}$, we first formulate the lower bound for Δt. Note that it is useless for the m obile sink to broadcast multiple times before it moves out of a sensor node's radio range, as all these broadcast messages will have the same hop counts. Hence, the first restriction is that two trail points should be separated by a distance longer than sensor node's average radio range $\overline{\tau}$, we have

$$\overline{\nu} \times \Delta t_m > \overline{\tau}. \tag{10}$$

Combining (9) and (10) makes $\Delta t > \frac{\overline{\tau}}{\overline{\nu}} + \Delta t_s$. In addition, for each transmission, the time duration should be long enough for a trail message to permeate the whole network, let this permeation time be φ, the lower bound of Δt is $\max\{\frac{\overline{\tau}}{\overline{\nu}} + \Delta t_s, \varphi\}$.

On the other hand, the upper bound of Δt is application specific. If the data gathering process is expected to finish in Ψ_{total} time, then during this time, the mobile sink should at least traversed all the d_v trail points. Therefore, we have $\Delta t < \frac{\Psi_{total}}{d_v}$. In summary, the range of Δt is given by

$$\Delta t \in \left[\max\left\{\frac{\overline{\tau}}{\overline{\nu}} + \Delta t_s, \varphi\right\}, \frac{\Psi_{total}}{d_v}\right). \tag{11}$$

Once Δt is determined, we can derive the step size parameter K as follows:

$$\begin{aligned} \Delta t_m \times \overline{\nu} &= K \times \overline{\tau} \\ &\Rightarrow \\ K &= \frac{\Delta t_m \times \overline{\nu}}{\overline{\tau}}. \end{aligned} \tag{12}$$

As this theoretical range of Δt is very broad and application specific, in Figs. 12a and 12b, we plotted some simulation results for a number of broadcast frequencies. The broadcasting frequency is indicated by the time interval between to consecutive broadcasts. We can see that shorter broadcast interval, i.e., more frequent control message broadcasting, does benefit the average route length, as trail references are refreshed in a timely fashion. However, higher update frequency propagates more messages, thereby incurring more energy consumption, especially for large network size. It is important to find a tradeoff point balancing different requirements when it comes to real application implementation.

Based on the conceptual sensitivity analysis in this section, choices of these parameters settings depend on specific application scenarios and user requirements. The analysis here can be used as a guideline for real system design, and can also be used as performance metrics for comparison study with other schemes.

6 Conclusion

We presented the SinkTrail and its improved version, SinkTrail-S protocol, two low-complexity, proactive data reporting protocols for energy-efficient data gathering. SinkTrail uses logical coordinates to infer distances, and establishes data reporting routes by greedily selecting the shortest path to the destination reference. In addition,

SinkTrail is capable of tracking multiple mobile sinks simultaneously through multiple logical coordinate spaces. It possesses desired features of geographical routing without requiring GPS devices or extra landmarks installed. SinkTrail is capable of adapting to various sensor field shapes and different moving patterns of mobile sinks. Further, it eliminates the need of special treatments for changing field situations. We systematically analyzed energy consumptions of SinkTrail and other representative approaches and validated our analysis through extensive simulations. The results demonstrate that SinkTrail finds short data reporting routes and effectively reduces energy consumption. The impact of various design parameters used in SinkTrail and SinkTrail-S are investigated to provide guidance for implementation.

We are currently working with collaborators in the GreenSeeker system [20]. Through one-hop sensing, the GreenSeeker system applies the precise amount of Nitrogen adaptive to spatial and temporal dynamics of the farmland, increasing yield and reducing Nitrogen input expense. The SinkTrail protocol can be further integrated with the GreenSeeker system to enable large-scale multihop sensing on demand and automate spray systems for optimal fertilizer and irrigation management.

ACKNOWLEDGMENTS

The work presented in this paper is supported in part by US National Science Foundation under Grant Nos. CNS-0709329, CNS-0923238, and CNS-0940805 (BBN subcontract) and OCAST.

REFERENCES

[1] S. Basagni, A. Carosi, E. Melachrinoudis, C. Petrioli, and Z.M. Wang, "Controlled Sink Mobility for Prolonging Wireless Sensor Networks Lifetime," *ACM/Elsevier Wireless Networks,* vol. 14, pp. 831-858, 2007.

[2] C. Chou, K. Ssu, H. Jiau, W. Wang, and C. Wang, "A Dead-End Free Topology Maintenance Protocol for Geographic Forwarding in Wireless Sensor Networks," *IEEE Trans. Computers,* vol. 60, no. 11, pp. 1610-1621, Nov. 2010.

[3] D. Coffin, D. Van Hook, S. McGarry, and S. Kolek, "Declarative Ad-Hoc Sensor Networking," *Proc. SPIE,* vol. 4126, p. 109, 2000.

[4] M. Demirbas, O. Soysal, and A. Tosun, "Data Salmon: A Greedy Mobile Basestation Protocol for Efficient Data Collection in Wireless Sensor Networks," *Proc. IEEE Third Int'l Conf. Distributed Computing in Sensor Systems,* pp. 267-280, 2007.

[5] K. Fodor and A. Vidács, "Efficient Routing to Mobile Sinks in Wireless Sensor Networks," *Proc. Third Int'l Conf. Wireless Internet (WICON),* pp. 1-7, 2007.

[6] R. Fonseca, S. Ratnasamy, J. Zhao, C.T. Ee, D. Culler, S. Shenker, and I. Stoica, "Beacon Vector Routing: Scalable Point-To-Point Routing in Wireless Sensornets," *Proc. Second Conf. Networked Systems Design and Implementation (NSDI),* pp. 329-342, 2005.

[7] Q. Huang, C. Lu, and G. Roman, "Spatiotemporal Multicast in Sensor Networks," *Proc. First ACM Conf. Embedded Networked Sensor Systems (SenSys),* pp. 205-217, 2003.

[8] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," *Proc. MobiCom,* pp. 56-67, 2000.

[9] M. Keally, G. Zhou, and G. Xing, "Sidewinder: A Predictive Data Forwarding Protocol for Mobile Wireless Sensor Networks," *Proc. IEEE Sixth Ann. Comm. Soc. Conf. Sensor, Mesh and Ad Hoc Comm. and Networks (SECON),* pp. 1-9, June 2009.

[10] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and Scalable Simulation of Entire Tinyos Applications," *Proc. First ACM Conf. Embedded Networked Sensor Systems (SenSys),* pp. 126-137, 2003.

[11] Z. Li, N. Wang, A. Franzen, P. Taher, C. Godsey, H. Zhang, and X. Li, "Practical Deployment of an In-Field Soil Property Wireless Sensor Network," *Computer Standards and Interfaces,* vol. 33, pp. 13-23, 2011.

[12] B. Liu, W. Ke, C. Tsai, and M. Tsai, "Constructing a Message-Pruning Tree with Minimum Cost for Tracking Moving Objects in Wireless Sensor Networks is Np-complete and an Enhanced Data Aggregation Structure," *IEEE Trans. Computers,* vol. 57, no. 6, pp. 849-863, June 2008.

[13] J. Luo and J.-P. Hubaux, "Joint Mobility and Routing for Lifetime Elongation in Wireless Sensor Networks," *Proc. IEEE INFOCOM,* vol. 3, 2005.

[14] M. Ma and Y. Yang, "Data Gathering in Wireless Sensor Networks with Mobile Collectors," *Proc. IEEE Int'l Symp. Parallel and Distributed Processing (IPDPS),* pp. 1-9, Apr. 2008.

[15] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring," *Proc. First ACM Int'l Workshop Wireless Sensor Networks and Applications (WSNA),* pp. 88-97, 2002.

[16] T. Moscibroda, R. O'Dell, M. Wattenhofer, and R. Wattenhofer, "Virtual Coordinates for Ad Hoc and Sensor Networks," *Proc. Joint Workshop Foundations of Mobile Computing (DIALM-POMC),* pp. 8-16, 2004.

[17] T. Park, D. Kim, S. Jang, S. eun Yoo, and Y. Lee, "Energy Efficient and Seamless Data Collection with Mobile Sinks in Massive Sensor Networks," *Proc. IEEE Int'l Symp. Parallel and Distributed Processing (IPDPS),* pp. 1-8, May 2009.

[18] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic Routing without Location Information," *Proc. MobiCom,* pp. 96-108, 2003.

[19] D. Shah and S. Shakkottai, "Oblivious Routing with Mobile Fusion Centers over a Sensor Network," *Proc. IEEE INFOCOM,* pp. 1541-1549, 2007.

[20] J. Solie, M. Stone, W. Raun, G. Johnson, K. Freeman, R. Mullen, D. Needham, S. Reed, C. Washmon, and P. Robert, "Real-Time Sensing and N Fertilization with a Field Scale GreenSeeker Applicator," *Proc. Am. Soc. Agronomy,* 2002.

[21] A.A. Somasundara, A. Ramamoorthy, and M.B. Srivastava, "Mobile Element Scheduling for Efficient Data Collection in Wireless Sensor Networks with Dynamic Deadlines," *Proc. IEEE 25th Int'l Real-Time Systems Symp. (RTSS),* pp. 296-305, 2004.

[22] A.A. Somasundara, A. Ramamoorthy, and M.B. Srivastava, "Mobile Element Scheduling for Efficient Data Collection in Wireless Sensor Networks with Dynamic Deadlines," *Proc. IEEE 25th Int'l Real-Time Systems Symp. (RTSS),* pp. 296-305, 2004.

[23] O. Soysal and M. Demirbas, "Data Spider: A Resilient Mobile Basestation Protocol for Efficient Data Collection in Wireless Sensor Networks," *Proc. Int'l Conf. Distributed Computing in Sensor Systems (DCOSS),* 2010.

[24] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, "A Two-tier Data Dissemination Model for Large-Scale Wireless Sensor Networks," *Proc. MobiCom,* pp. 148-159, 2002.

[25] F. Ye, G. Zhong, S. Lu, and L. Zhang, "Gradient Broadcast: A Robust Data Delivery Protocol for Large Scale Sensor Networks," *Wireless Networks,* vol. 11, no. 3, pp. 285-298, 2005.

[26] M. Younis, M. Youssef, and K. Arisha, "Energy-Aware Routing in Cluster-Based Sensor Networks," *Proc. IEEE 10th Int'l Symp. Modeling, Analysis and Simulation of Computer and Telecomm. Systems (MASCOTS),* pp. 129-136, 2002.

[27] L. Yu, N. Wang, and X. Meng, "Real-Time Forest Fire Detection with Wireless Sensor Networks," *Proc. Int'l Conf. Wireless Comm., Networking and Mobile Computing,* vol. 2, pp. 1214-1217, 2005.

[28] M. Zhao, M. Ma, and Y. Yang, "Mobile Data Gathering with Space-Division Multiple Access in Wireless Sensor Networks," *Proc. IEEE INFOCOM,* pp. 1283-1291, Apr. 2008.

[29] M. Zhao, M. Ma, and Y. Yang, "Efficient Data Gathering with Mobile Collectors and Space-Division Multiple Access Technique in Wireless Sensor Networks," *IEEE Trans. Computers,* vol. 60, no. 3, pp. 400-417, Mar. 2011.

[30] M. Zhao and Y. Yang, "Bounded Relay Hop Mobile Data Gathering in Wireless Sensor Networks," *IEEE Trans. Computers,* vol. 61, no. 2, pp. 265-277, Feb. 2012.

Xinxin Liu received the BE degree in software engineering from Jilin University, China. Currently, she is working toward the PhD degree in the Department of Computer and Information Science and Engineering at the University of Florida. Her research interests include wireless sensor networks and resource management. She is a student member of the IEEE.

Han Zhao received the BE degree in software engineering from Jilin University, China, and the MSc degree in computer science from Oklahoma State University. Currently, he is working toward the PhD degree in the Department of Computer and Information Science and Engineering at the University of Florida. His research interests include parallel and distributed computing, autonomic systems, and P2P systems. He is a student member of the ACM and the IEEE.

Xin Yang received the BS degree in the Department of Computer Science and Technology from Nanjing University, China. Currently, he is working toward the PhD degree in the Department of Computer and Information Science and Engineering at the University of Florida. His research interests include high-performance computing and cloud computing.

Xiaolin Li received the PhD degree in computer engineering from Rutgers University. Currently, he is working as an associate professor in the Department of Electrical and Computer Engineering at the University of Florida. His research interests include parallel and distributed systems, cyber-physical systems, and network security. His research has been sponsored by US National Science Foundation (NSF), Department of Homeland Security (DHS), and other funding agencies. He is an associate editor of several international journals and a program chair or cochair for more than 10 international conferences and workshops. He is on the executive committee of IEEE Technical Committee on Scalable Computing (TCSC) and served as a panelist for NSF. He is directing the Scalable Software Systems Laboratory (http://www.s3lab.ece.ufl.edu). He received the National Science Foundation CAREER Award in 2010. He is a member of the IEEE and the ACM.