2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware

TwinRoute: Energy-Efficient Data Collection in Fixed Sensor Networks with Mobile Sinks

Ricklef Wohlers [#1], Niki Trigoni [#2], Rui Zhang [*3], Stephen Ellwood [+4]

[#]*Computer Laboratory, University of Oxford*
Wolfson Building, Parks Road, Oxford, OX1 3QD, UK
[1]ricklef.wohlers@comlab.ox.ac.uk
[2]niki.trigoni@comlab.ox.ac.uk

[*]*IBM Almaden Research Center*
[3]ruizhang@us.ibm.com

[+]*Department of Zoology, University of Oxford*
Tubney House, Abingdon Road, Abingdon, OX13 5QL, UK
[4]stephen.ellwood@zoo.ox.ac.uk

***Abstract*—Collecting data from source sensor nodes to designated sinks is a common and challenging task in a wide spectrum of sensor network applications, ranging from animal monitoring to security surveillance. A number of approaches exploiting sink mobility have been proposed in recent years: some are proactive, in that sensor nodes push their readings to storage nodes from where they are collected by roaming mobile sinks, whereas others are reactive, in that mobile sinks pull readings from nearby sensor nodes as they traverse the sensor network. In this paper, we point out that deciding which data collection approach is more energy-efficient depends on application characteristics, including the mobility patterns of sinks and the desired freshness of collected data. We illustrate cases where combining proactive and reactive modes of data collection is particularly beneficial. This motivates the design of TwinRoute, a novel hybrid algorithm that can flexibly mix the two collection modes at appropriate levels depending on the application scenario. Our extensive experimental evaluation, using synthetic and real-world network topologies and sink traces, shows that TwinRoute outperforms the pure approaches, achieving desirable tradeoffs between energy expenditure and timely delivery of sensor data.**

I. Introduction

Sensor network applications are numerous and diverse, ranging from wild animal monitoring to security surveillance. They typically include a large number of battery-powered sensor nodes that monitor ambient environmental conditions, and communicate their readings, hop by hop, to one or more sinks. A major performance constraint in sensor networks is energy, a significant portion of which is spent on communications. When sinks are static, the sensor nodes closest to each sink carry a disproportionate communication load as they are tasked to relay messages from other parts of the network. Conversely, using mobile sinks to collect data reduces the overall data forwarding cost, balances the communication load, and thus prolongs the network lifetime.

In this paper, we consider applications where mobile sinks with abundant energy, storage and processing resources, already exist in the deployment environment. Mobile sinks may be human, vehicular or other mobile objects capable of carrying a PDA-type device. They can wirelessly communicate with static sensor nodes within range to collect data from the network. Their movement is not controlled, and they do not need know their location in the network. Sink mobility patterns vary widely across different applications. Some scenarios involve only mobile sinks, whereas others feature a mixture of mobile and fixed sinks; mobile sinks may follow completely random trajectories, or repeatedly take predictable paths.

Similarly, applications vary in their requirements for data freshness - i.e. the time between data generation at a sensor node and delivery to a user. Emergency or intrusion detection applications require sensor events to be delivered within a few seconds from their detection, whereas traffic monitoring applications can tolerate delivery delays of 10-20 minutes. Variations in desirable data freshness are often observed within the scope of the same application. For example, in a wildlife monitoring application, an animal movement event may need to be reported immediately in order to activate a nearby camera and capture its behaviour; delays of several minutes can be tolerated to activate environmental sensors to capture microclimatic conditions in the animal's vicinity; delays of hours are tolerated for monitoring long-term animal behaviour.

Previous work on data collection from fixed sensors to mobile sinks has focused on two extreme approaches: 1) A pure *proactive* approach, in which sensor nodes proactively forward their data to fixed storage nodes, from where data are collected opportunistically by visiting mobile sinks. 2) A pure *reactive* approach, in which sensor nodes in the vicinity of mobile sinks react to sink-initiated probes for data collection. The proactive approach requires careful selection of fixed storage nodes, but, once storage nodes are selected, relatively stable routes are used from sensor nodes to storage nodes (subject to node and link failures). The reactive approach, however, requires frequent route updates from fixed sensor nodes to mobile sinks.

The aim of this paper is to address the following questions: Given a specific application scenario, which is the most energy-efficient data collection scheme? Is this dependent on the mobility patterns of mobile sinks, or the application

978-0-7695-3650-7/09 $25.00 © 2009 Crown Copyright
DOI 10.1109/MDM.2009.30

requirements for data freshness? Are there scenarios where combining the two pure approaches into a hybrid scheme could yield a more energy-efficient way of data collection? If so, how could we mix the two approaches in a flexible way so that we can strike a good balance between the proactive and reactive elements? Our specific research contributions are as follows:

- We study different flavors of pure proactive and reactive schemes for data collection, which are inspired by previous work [1], [2], and discuss their shortcomings.
- We evaluate proactive and reactive schemes in a variety of scenarios, varying the sink mobility patterns and the user requirements for data freshness. We show that the proactive approach is more energy-efficient than the reactive approach in some cases and worse in others.
- We propose TwinRoute, a novel algorithm that combines the benefits of the two pure schemes. TwinRoute can be easily tuned to blend proactive and reactive elements at appropriate levels depending on the application.
- We provide an extensive experimental evaluation of TwinRoute, using synthetic and real network topologies and sink traces, and compare them with the competing pure schemes. We show that TwinRoute is always at least as good as the best of the pure approaches, and we identify classes of applications where TwinRoute significantly outperforms both of them.

This paper is organised as follows: Section II presents the assumptions of our model and the objective of our work; Section III provides detailed descriptions of the pure schemes, and the novel TwinRoute algorithm; Section IV evaluates the performance of the three approaches in a variety of scenarios using synthetic and real sink mobility traces; Section V provides an overview of related work; Section VI presents our conclusions and ideas for future work.

II. Problem Definition

Consider a wireless network consisting of n static sensor nodes deployed in an area of interest, and m mobile sinks roaming through the area. Static sensor nodes are typically battery-powered and therefore energy-constrained. They sense the environment and generate data that needs to be forwarded to one of the mobile sinks. We assume that mobile sinks are not energy-constrained, and once data arrives at a mobile sink, it becomes available to the application users. In this paper, we do not assume control over the mobile sink's movement. As a result, our work is applicable to a potentially much wider set of applications than those targeted by previous work [2], [3], [4], [5].

In principle, data freshness requirements can vary not only across applications, but also across different packets of the same application. A data packet is considered to be fresh when it arrives at a mobile sink within a user-specified freshness threshold, which depends on the urgency of the encoded data. Each application defines a lower bound on the delivery ratio of fresh packets. For example, a habitat monitoring application may require that animal observations have freshness thresholds of 1 minute, whereas temperature change observations have freshness thresholds of 10 minutes. It also requires that 99% of all packets must be delivered within their specified freshness thresholds.

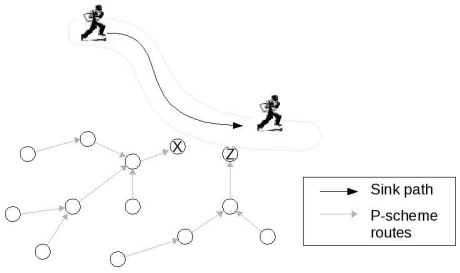


Fig. 1. P-scheme in action in an example sensor network where nodes X and Z have been selected as storage nodes

Given specific application requirements for data freshness, our objective is to devise energy-efficient algorithms for data collection from fixed sensor nodes to mobile sinks. Since communication typically dominates the total energy expenditure, our goal is to design distributed algorithms that minimise the total number of packet transmissions within the fixed sensor network.

III. The TwinRoute algorithm

In this section, we propose *TwinRoute*, a hybrid approach to forwarding data from static sensor nodes (from hereon in referred to as *sensors*) to mobile sinks that combine two approaches: i) a *proactive* scheme in which sensors proactively push data to carefully selected storage nodes, and from where data is collected by roaming sinks, and ii) a *reactive* scheme in which mobile sinks continually advertise their presence to nearby sensors, and the latter react by forwarding their data back to the sinks. A detailed description of the two pure schemes and the proposed TwinRoute algorithm is provided below.

A. Proactive scheme (P-scheme)

The P-scheme is a generalisation of existing algorithms that build and maintain data dissemination trees rooted in predefined storage nodes [1], [2], [6]. These existing algorithms typically assume fixed or highly repeatable sink trajectories along which storage nodes are selected. In contrast, we consider the more general case in which mobile sinks traverse the network in a more random fashion on an individual basis, yet collectively exhibit a regular visiting pattern. By keeping a record of sink visits, it is possible to identify sensors that are likely to deliver data to a sink within a given freshness threshold and use them as storage nodes. The P-scheme selects storage nodes based on the history of sink visits, and builds dissemination trees rooted at them, thus enabling the remaining sensors to proactively forward their data to the closest storage node (see figure 1). More specifically, the P-scheme is a localised algorithm that involves four tasks, as shown in the pseudocode depicted in Table I.

The first task (P1) involves keeping track of sink visits: when a sink is detected within range, the node updates a

INPUT VARIABLES:
SDT - storage delay threshold;
ID - node identity;
LOCAL VARIABLES:
$distP$ - number of hops to storage node;
$parentP$ - parent node ID in routing tree;
$visits$ - list of timestamps of sink visits;
$storageDelay$ - delay for data stored at storage node;

MAIN ALGORITHM:
$distP = \infty; parentP = ID; visits = \emptyset;$
WHILE true
[P1] IF mobile sink comes in range
$visits = visits \bigcup \{currentTime\};$

[P2] compute $storageDelay$ using $visits$;
IF $storageDelay < SDT$
$distP = 0$; $b.dist = 0$; $b.source = ID$;
broadcast b;

[P3] IF a P beacon b is received
IF $b.dist < distP$
$distP = b.dist$; $parentP = b.source$;
$b.dist = b.dist + 1$; $b.source = ID$;
broadcast b;

[P4] IF any data **AND** sink in range
send *all* data to sink;
ELSE IF any data **AND** $parentP \neq ID$
send *all* data to $parentP$;

TABLE I
PSEUDOCODE OF P-SCHEME.

local list of recent sink visit times. The second task (P2) involves a storage node selection scheme. Let *storage delay threshold* (SDT) denote an upper bound on the time that packets should wait at storage nodes before being delivered to mobile sinks. If, based on recent information about sink visits, a node expects to deliver the majority (say 85%) of buffered packets within $storageDelay$, and $storageDelay < SDT$, the node becomes a storage node and broadcasts a beacon to initiate tree construction.

The third task (P3) describes how a sensor processes a tree construction beacon upon hearing it, so that it associates itself to a route leading to its closest (in terms of hops) storage node. The final task (P4) presents a zero-wait, push data reporting mechanism that sends data along the established routes to the closest storage node, or directly to the sink if the latter is in range.

The storage delay threshold is a tunable parameter of P-scheme, and its value must be defined taking into account the application requirements for data freshness. The higher the SDT the more nodes elect themselves as storage nodes and the smaller the communication trees formed around them. This means that high SDT values result in small communication cost at the expense of low data freshness. Conversely, low SDT values result in high data freshness, at the expense of high communication cost. Thus, by carefully varying SDT, the P-scheme gracefully trades communication cost for data freshness.

INPUT VARIABLES:
TD - reactive routing tree depth (number of hops);
ID - node identity;
LOCAL VARIABLES:
$distR$ - number of hops to sink;
$parentR$ - parent node ID in routing tree;
$justCovered$ - flag indicating recent tree coverage;

MAIN ALGORITHM:
$distR = \infty$; $parentR = rootR = ID$;
$justCovered = false$;
WHILE true
[R1] IF mobile sink comes in range
IF $TD > 0$ **AND** $justCovered == false$
$distR = 0$; $b.dist = 1$; $b.source = ID$;
broadcast b;
$justCovered = true$;
start $treeDestructTimer$;
start $justCoveredTimer$;

[R2] IF a R beacon b received
// update R-scheme
IF $b.dist < TD$ **AND** $justCovered == false$
$distR = b.dist$; $parentR = b.source$;
$justCovered = true$;
start $justCoveredTimer$;
start $treeDestructTimer$;
// broadcast tree if next hop smaller than TD
IF $b.dist + 1 < TD$
$b.dist = b.dist + 1$; $b.source = ID$
broadcast b

[R3] IF $treeDestructTimer$ expires
$distR = \infty$; $parentR = ID$;

[R4] IF $justCoveredTimer$ expires
$justCovered = false$;

[R5] IF any data **AND** sink in range
send *all* data to sink;
ELSE IF any data **AND** $parentR \neq ID$
send *all* data to $parentR$;

TABLE II
PSEUDOCODE OF R-SCHEME.

B. Reactive scheme (R-scheme)

An implicit assumption behind the proactive scheme presented in the previous subsection is that there is some *predictable* long-term trend in the collective behaviour of sink trajectories. In particular, the P-scheme exploits the fact that the frequency of mobile sink visits at a sensor can be forecast from recent history. As a result, a constantly changing network, e.g. one with alternating periods of high and low sink traffic, can significantly undermine the effectiveness of this scheme. Worse, some sensor networks may experience periods of random sink trajectories (e.g. visitors to a national park) followed by periods of highly repeatable sink trajectories (e.g. scientists studying animal species in a national park). In this case, it may be difficult to discover a consistent visit history to make accurate predictions. Wary of the above situations we present R-scheme, a reactive algorithm that does not presume long-term statistics and reacts to the movement of individual sinks in real-time, it being inspired by [7]. As described in Table II, the R-scheme is a localised algorithm consisting

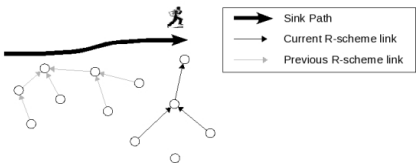


Fig. 2. R-scheme in action with $TD = 2$ in an example sensor network.

of five tasks. In the first task (R1), each sensor detects a mobile sink as it comes in range and subsequently initiates a routing-tree construction process. In the second task (R2), nodes handle routing-tree construction beacons received from their neighbours. This process is restricted so that 1) no tree construction beacon exceeds a tree depth of TD hops and 2) if a sensor has recently participated in another routing-tree rooted at a previous position of the sink, it does not participate in a new tree construction until the $justCoveredTimer$ expires. The purpose of the first restriction is to constrain the depth of reactive trees and thus the energy consumed in constructing them and in delivering data along their paths. The second restriction follows the intuition that it is not worth extending the reactive tree to an area where there is little data left to report. The third task (R3) automatically destroys the current tree after a short interval that the mobile sink typically takes to move out of range of the tree's root[1]. The fourth task (R4) allows nodes to participate again in tree construction after a sufficient time interval has elapsed. Finally, in the fifth task (R5), nodes forward buffered data to their parent in the reactive tree, or to the sink in case the latter is within range.

Figure 2 depicts how reactive routing works in an example sensor network, where a mobile sink traverses node A, B, C and D. In particular, the figure depicts the moment when a new tree rooted at D is being constructed, and serves to demonstrate the customisations described in the previous paragraph. Nodes B, C and E do not participate in the new tree rooted at D, as they were very recently part of the tree rooted at A ($justCoveredTimer$). For the same reason, B and C did not become root nodes when visited by the sink. In addition, tree construction beacons are propagated up to nodes F and G, and they are not forwarded to H, as the maximum tree depth ($TD = 2$) has been reached.

The tree depth (TD) is a tunable parameter of R-scheme, and its value must be decided based on application requirements for data freshness. The higher the TD the higher the data freshness at the expense of higher communication cost. Conversely, the lower the TD the lower the communication cost at the expense of lower data freshness. Thus, by carefully varying TD, the R-scheme gracefully trades communication cost for data freshness.

[1]For simplicity, we assume that this interval is the same for all nodes in the tree, and it suffices to empty their buffers and forward any data to the sink before it moves out of range. This is realistic in applications where the sink does not move very fast, and routing trees are relatively shallow.

C. Hybrid approach (TwinRoute)

In the previous subsections we presented a proactive and a reactive algorithm for data collection. In particular, we showed that the P-scheme is designed to exploit long-term historical information about sink visits and elect relatively stable storage nodes to which sensors proactively push their data. In contrast, the R-scheme is designed to react to real-time sink movements and allows a sink to pull data from nearby sensors in its passage from their neighbourhood. We are now in a position to present *TwinRoute*, a hybrid algorithm that combines the proactive and reactive data collection techniques with the aim of taking advantage of each of their respective strengths.

TwinRoute is essentially a parallel merge of P-scheme and R-scheme. In other words, each sensor maintains both the sink visit history required by proactive routing, as well as tracks each sink as it comes in and out of range as specified by reactive routing. In addition, each sensor also listens to and broadcasts beacons for both schemes. As a result, a network can exploit both relatively stable and large routing trees created by P-scheme, and relatively transient and small routing trees created by the R-scheme. Each sensor is thus always part of a proactive tree, whilst it is occasionally presented with the choice to dynamically participate in a reactive tree whenever a mobile sink appears in its vicinity.

Typically, great advantages can be gained over a pure proactive scheme when a mobile sink unexpectedly appears near sensors that are many hops away from the closest storage node. In TwinRoute, sensors have the opportunity to temporarily switch to the reactive routing tree and offload their data on much shorter paths. On the other hand, TwinRoute will demonstrate its superiority over a pure reactive scheme as long as there are areas in the network where mobile sink visits are consistently concentrated. Since we can be confident that mobile sinks will always revisit these areas in the future, there is little need to demolish routes once they are established (as in a pure reactive scheme).

TwinRoute reuses most parts of the R-and P-schemes to handle mobile sink visits and tree construction beacons. A critical change is made in the way R beacons are disseminated to build reactive trees. An additional condition $b.dist < distP - SP$ is enforced in the tree construction step of R-scheme (step $R2$), where SP is a scheme preference parameter that regulates when one of the pure schemes overwrites the other. When $SP > 0$, a sensor becomes part of reactive tree only if its distance to the sink on that tree is shorter (by SP hops) than its distance to a P-scheme storage node ($distP$). That is, the scheme preference value SP denotes the distance gain necessary to switch from P-scheme to R-scheme.

Figure 3 showcases the critical process of a switch to R-scheme when $SP = 2$. As a mobile sink travels by, node A becomes the root of a reactive tree and broadcasts R beacons. Nodes B, C and D are currently part of two separate P trees. The beacons fire the scheme switching condition on nodes C and D but not on node B.

The second subtle point in combining P-scheme and R-

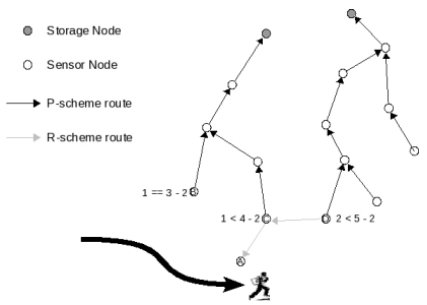


Fig. 3. TwinRoute in action with $SP = 2$ in an example sensor network.

scheme concerns the data-forwarding step. Once a sensor has switched to become part of a reactive tree, it always favours R-scheme when reporting data. All buffered data are sent along branches of the reactive tree, if one exists, to take full advantage of the existing routes to the sink. In the absence of a reactive tree, a sensor returns to using the routes of its proactive tree. However, unlike the pure P-scheme, data is not forwarded in a zero-wait mode, but in a delay-tolerant manner. Not forwarding data items as soon as they are generated at the source node ensures that there will be some data to capitalise on a switch to R-scheme should it happen. In order to do this, when storage nodes initiate tree construction around them (in the P-scheme), they include their storage delays in the tree construction beacons. The time that a message can be buffered locally at the source node must not exceed its data freshness threshold, minus the storage delay at the P-scheme storage node.

Since TwinRoute combines the proactive and reactive schemes, it inherits two important parameters from these schemes: 1) the storage delay threshold (SDT) used to determine if a sensor node elects itself to be a storage node, which is inherited from P-scheme, and 2) the tree depth (TD) of reactive trees formed dynamically around roaming sinks, which is inherited from R-scheme. By carefully tuning these two parameters, together with the scheme preference parameter (SP), we can come up with different flavors of TwinRoute, to suit the requirements of a particular application. In extreme cases, TwinRoute can completely suppress one of the two pure schemes, thus retiring to the other pure scheme. In the general case, TwinRoute can be tuned to strike the optimal balance of proactive and reactive routing, i.e. the one that minimises communication given application requirements for data freshness. This is demonstrated in the extensive evaluation of TwinRoute and pure schemes in Section IV.

IV. Experimental Evaluation

In this section, we provide an extensive evaluation of TwinRoute and competing pure schemes. Our key performance metric is *communication cost*, which we measure as the total number of packets sent by fixed sensor nodes in the process of data collection. In addition, we measure the delivery ratio of fresh packets; this is defined as the number of packets delivered to sinks within user-defined freshness thresholds divided by the total number of packets generated in the network. The algorithms were simulated in TOSSIM [8], and a series of experiments were run to measure their performance in a variety of conditions.

More specifically, in Section IV-A, we assess the performance of various flavours of P-scheme, R-scheme and TwinRoute, as we vary their parameters. The main goal is to understand the impact of their parameter values on their communication cost and delivery ratio of fresh packets. In Section IV-B, we investigate the impact of varying the mobility patterns of sinks on the performance of the three algorithms. In Section IV-C we assume that applications define freshness requirements that consist of 1) freshness thresholds for different packets and 2) a desirable delivery ratio of fresh packets. Our goal is to explore the impact of application requirements for freshness on the performance of TwinRoute and pure schemes. Finally, Section IV-D compares the algorithms using real-life sink traces (from animal movements) and a realistic sensor deployment designed by zoologists in our WildSensing project.

A. Impact of parameter values on algorithm performance

In this subsection, we study how by varying the parameters of the three algorithms, we can strike different tradeoffs between communication cost and delivery ratio of fresh packets. To this end, we fix the network topology, sink mobility and packet freshness threshold. More specifically, we simulate a network of 100 nodes deployed in a grid-like manner, spaced apart by 8 meters, and communicating with a 10 meter range (to simulate dense canopy environments, which are common in our WildSensing project). We simulate three mobile sinks moving according to the random way-point model at a fixed speed of 2m/s. We also set the packet freshness threshold to 50, 100 and 250 seconds, and run simulations for 1000 seconds.

Figures 4a-4d show that the performance of P-scheme and R-scheme is significantly impacted by their parameter values. Let us observe the impact of varying the storage delay threshold (SDT) parameter on the performance of P-scheme. When SDT is 0 secs, none of the sensor nodes assumes the storage role, because none of them can guarantee zero delays until the next sink visit. Not knowing of any storage nodes, sensors resort to buffering packets locally, until sinks come within range. As SDT increases beyond 100 secs, an increasing number of nodes become storage nodes, proactive trees around storage nodes become shorter, and the communication cost of P-scheme decreases. The communication savings, however, are counter-balanced by a noticeable fall in the delivery ratio of fresh packets. Figure 4c and 4d show a similar tradeoff between communication cost and delivery ratio in the case of R-scheme. As the tree depth of reactive trees increases, the delivery ratio of fresh packets increases at the expense of higher communication cost.

Figure 4a and 4c show that the communication cost of P-scheme and R-scheme depends purely on the values of their parameters (SDT and TD respectively). The delivery ratio, however, also depends on the requirement for packet freshness

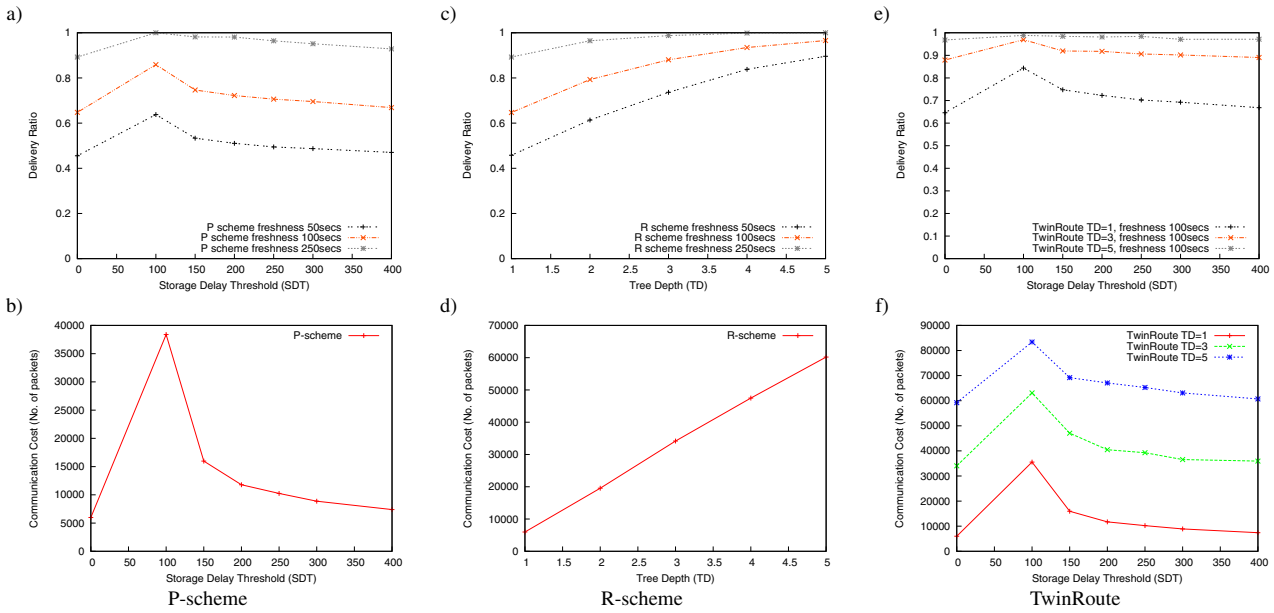


Fig. 4. Performance of P- and R-scheme as we vary their parameters

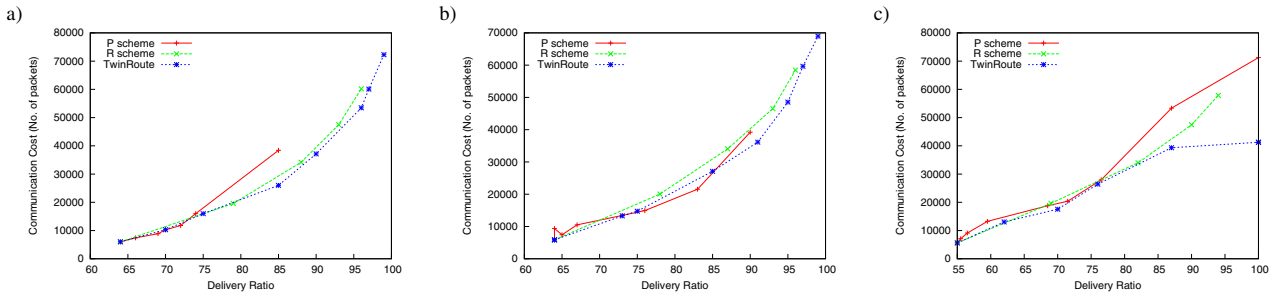
(Figure 4b and 4d). As expected, in both P- and R- schemes, the delivery ratio of fresh packets decreases as the freshness bound tightens from 250 to 50 secs.

Figure 4e and 4f show the impact of storage delay threshold (SDT) and reactive tree depth (TD) on the performance of TwinRoute, when the packet freshness threshold is set to 100 secs. Note that changing the values of one of the two parameters (say, SDT) does not have such a dramatic effect on the performance of TwinRoute, as it had on the performance of the corresponding pure scheme, due to the presence of the other pure scheme. TwinRoute, however, makes a qualitatively similar tradeoff between communication cost and delivery ratio, as observed in the pure schemes. In particular, as tree depth decreases and storage delay threshold increases, the communication cost of TwinRoute is reduced at the expense of delivering fewer packets within their freshness thresholds.

B. Impact of sink mobility

In the previous subsection, we showed that we can derive different versions of P-scheme, R-scheme and TwinRoute by varying their parameters. Different versions vary in how they trade communication cost for data freshness. Given specific application requirements for data freshness (e.g., packets must be delivered to sinks within 100 secs, and at least 95% of the total packets must be collected within this threshold), we can easily identify the most communication-efficient versions of P-scheme, R-scheme and TwinRoute that satisfy these requirements. We then compare the best versions of the three schemes in the context of different sink mobility models.

1) *Random mobility scenario:* three sinks moving according to the random waypoint model (RWP) at a fixed speed of 2m/s. The scenario is representative of cases where the mobile sinks do not embark on a purposeful journey and their movement is entirely unpredictable. Figure 5a shows the best performing P-, R- and hybrid schemes for different desired delivery ratios. Observe that R-scheme performs increasingly better than P-scheme, especially as applications require higher delivery ratios of fresh data (a packet is fresh if it is delivered within 100 secs). Note that P-scheme cannot achieve delivery ratios greater than 85% because none of the storage nodes are visited often enough by mobile sinks. TwinRoute puts more weight on the reactive element, and has comparable performance to R-scheme.
2) *Regular path scenario:* one sink follows a fixed path whilst two sinks move using the random waypoint model. In this case, P-scheme takes advantage of the increased predictability and outperforms R-scheme. However, P-scheme can achieve up to 90% delivery ratio, due to the limited frequency in which sinks visit storage nodes, whereas R-scheme can achieve higher delivery ratio, simply by using a higher tree depth. Again, in this case, TwinRoute performs similarly to the best of the two pure schemes.
3) *Static gateway scenario:* one static gateway and two sinks following a random waypoint mobility pattern. This reflects scenarios where sensor nodes have the option of forwarding their data to a static gateway or to roaming mobile sinks. In this case, the three schemes perform comparably when the desired delivery ratio is lower than 75%. R-scheme outperforms P-scheme for delivery ratios between 75% and 95% because the proactive tree to a single fixed gateway is very deep, compared to short reactive trees formed around mobile sinks. Unlike the previous two scenarios, P-scheme achieves very high delivery ratios, close to 100%, due to the



3 RWP sinks | 1 regular path sink, 2 RWP sinks | 1 static gateway, 2 RWP sinks

Fig. 5. Best communication cost for Proactive, Reactive and TwinRoute algorithm for various desired delivery ratios

fixed gateway. Such high delivery ratios are unachievable by R-scheme which collects data up to 5 hops away from sinks. When high delivery ratios close to 100% are required, TwinRoute, significantly outperforms both pure schemes.

Overall, we conclude that the algorithm performance is highly dependent both on sink mobility and on desired delivery ratio of fresh packets. R-scheme is preferred to P-scheme when sink movement is very unpredictable and applications require high delivery ratios. P-scheme outperforms R-scheme when sinks visit storage nodes frequently enough (or all the time as in the static gateway scenario). TwinRoute builds upon the strengths of the pure schemes: it typically performs at least as well as the best of them, and in certain special cases, it outperforms both.

C. Impact of data freshness distribution

In Section IV-B, we assumed that all packets of an application have the same freshness threshold (e.g. 100 secs). In this section, we extend our study to applications that feature a mix of time-critical and delay-tolerant packets. For example, in a farm monitoring application, 20% of packets concern animals trying to cross the farm boundaries, and must be reported to staff within seconds. The remaining 80% of packets concern animal contact events and need to be reported within minutes. Similarly, a traffic monitoring application may require imminent collection of accident-related data, and slightly more delay-tolerant collection of car traffic bottleneck events.

The table below includes the freshness requirements of four different application scenarios that we use to evaluate TwinRoute and the pure schemes: columns indicate the packet freshness thresholds, whereas rows denote the required delivery rate of fresh packets.

	Data Freshness	
Delivery Ratio	100 seconds	80% 250seconds, 20% 50 seconds
75%	wildlife boundary monitoring	farm monitoring
95%	accident monitoring	traffic monitoring

Let us now measure the performance of the best versions of P-scheme, R-scheme and TwinRoute in these four applications. Figure 6 considers the application scenarios that tolerate a freshness of 100 seconds for their packets, and require 75% (left graph) and 95 % (right graph) delivery ratio. TwinRoute is shown to perform better than the pure schemes, and its benefits are more pronounced in the case that the application has a fixed gateway and requires a high delivery ratio (95%).

Figure 7 shows the performance of the three schemes in applications that have two types of packets with two different freshness thresholds. Again, the benefits of TwinRoute are more evident in the application that requires a high delivery ratio of fresh packets (right graph). Comparing the right graphs of Figures 6 and 7, one can see that TwinRoute is particularly efficient in delivering packets with different freshness thresholds. The reason is that it combines a proactive approach to deliver delay-tolerant packets with a reactive approach to deliver time-critical packets. Note also that in many cases, P-scheme and R-scheme cannot even achieve the desired delivery ratio (as denoted by the absence of corresponding bars from the graphs).

D. Real deployment scenario

In this section, we evaluate the hybrid and pure data collection schemes using real traces followed by zoologists in Wytham Fields, Oxford, UK, and a realistic sensor layout designed by zoologists to monitor badgers in the same area (sensors are positioned close to the setts and latrines of a badger community). The traces were recorded by zoologists roaming the area over a period of 7 days during which data was typically collected in the morning and the evening. The zoologists did not visit exactly the same sites each time they traversed the area. The traces loosely follow given paths and thereby share some common focal points, to constitute sink mobility in-between random walk and the static gateway scenario (see Figure 8).

The sensor deployment consists of 34 nodes spread over an area of 1000x1000 meters with a communication range of 200 meters. The packet freshness threshold is set to 48 hours; this was a reasonable value given that very few nodes are visited by mobile sinks more often than 36 hours. Each sensor node generates a packet of sensor data (e.g. temperature readings, badger sightings) every five minutes.

The results depicted in Figure 9 show that TwinRoute has a similar performance to the pure schemes when 80% of packets

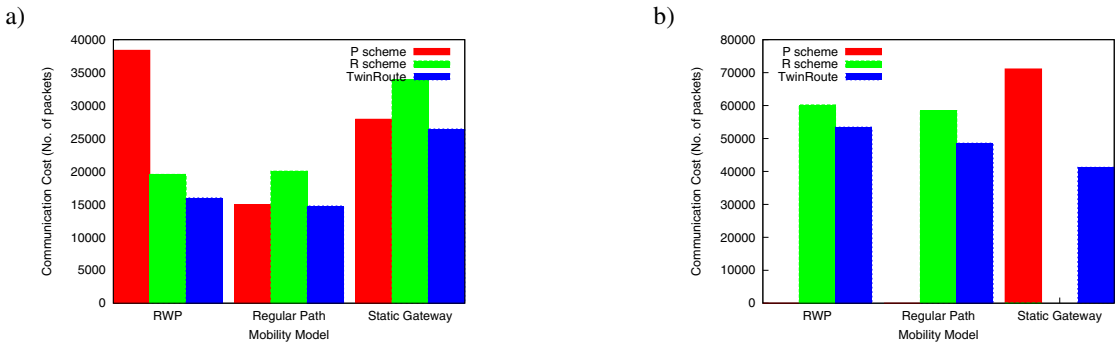


Fig. 6. Best communication cost for P- and R-scheme and TwinRoute for various mobility models, all packets have the same freshess of 100 seconds. One application requires 75% delivery ratio (left graph), and the other requires 95% delivery ratio (right graph)

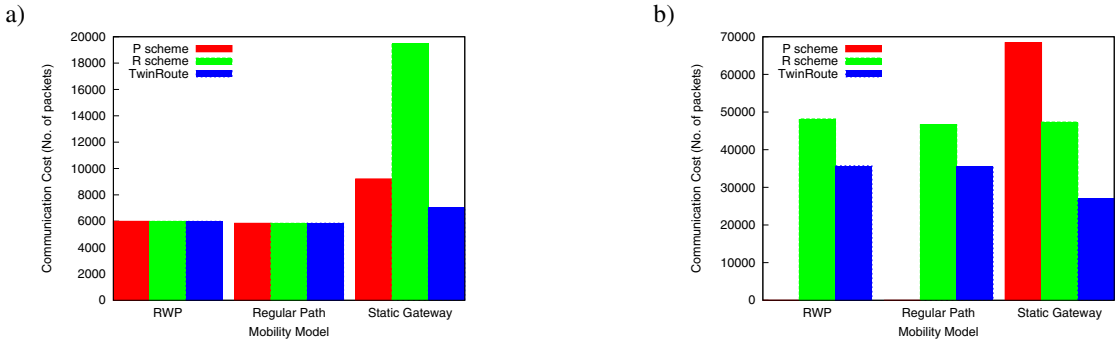


Fig. 7. Best communication cost for P- and R-scheme and TwinRoute for various mobility models, 20% of packets have freshness 50 secs and 80% of packets have freshness 250 secs. One application requires 75% delivery ratio (left graph), and the other requires 95% delivery ratio (right graph)

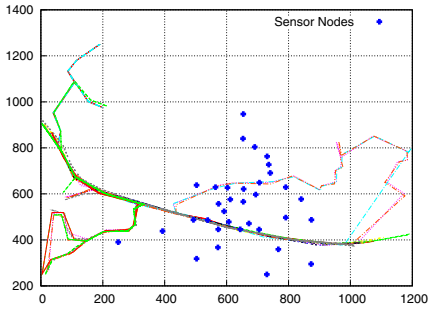


Fig. 8. Real-life sensor network topology and sink traces

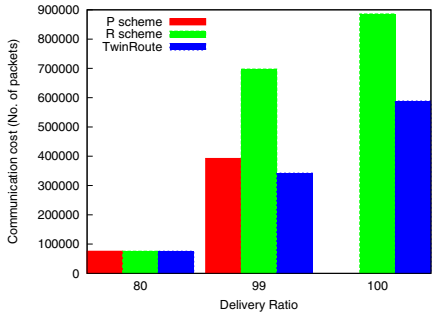


Fig. 9. Real-world simulation results for varying reliabilities

must be delivered within their threshold of 48 hours. However, if the habitat monitoring application required *all* packets to be delivered within their threshold, TwinRoute would significantly outperform its pure counterparts. To conclude, TwinRoute is particularly useful when very high delivery ratios are required. Recall that this result agrees with our previous results of testing the three schemes using synthetic sink traces in Sections IV-B and IV-C.

V. Related Work

There has been a plethora of work on energy-efficient dissemination of sensor data from static sensor nodes to one or more mobile sink nodes. Depending on the assumption about delay tolerance, we distinguish between real-time (uninterrupted or very low delay) data collection, and delay-tolerant data collection.

Real-time (Low-delay) Data Collection requires data sources to be continuously connected to the sinks via multi-hop routes. Luo et al. proposed Two-Tier Data Dissemination (TTDD) [9], a grid-like approach in which *dissemination* nodes, which are closest to the grid points, perform high-tier query and data routing, whereas low-tier routing is used for communication between dissemination nodes and mobile sinks. HCDD [10] is another hierarchical cluster-based approach to forwarding data from sources to mobile sinks, which, unlike TTDD, does not require sensor nodes to be location aware. Both TTDD and HCDD are pure reactive algorithms, since they require sinks to register with data sources.

The SEAD protocol (Scalable Energy-Efficient Asyn-

chronous Dissemination) [11] builds and maintains a data collection tree rooted at the data source, to which mobile sinks establish a connection by selecting a nearby *access* node that manages the connection on their behalf. The SEAD-protocol is shown to be more energy-efficient than directed diffusion [12], TTDD [9] or ADMR [13]. Unlike our work, which aims to propagate data from multiple sources to one of the mobile sinks, SEAD is designed to forward data from one data source to multiple mobile sinks. It is a reactive protocol in that sinks send *join* queries to the source to connect and receive data along the dissemination tree.

Unlike the SEAD-protocol, that makes no assumption about the mobile sink trajectory, the learning-based approach proposed by Baruah et al. [14] assumes the mobile sink to have a predictable path[2]. Nodes use a combination of statistical and reinforcement learning to come up with good delivery paths to the mobile sink. This approach is similar to the P-scheme described in this paper, in that it exploits inherent patterns of sink mobility that are sustained for a significant period of time. However, unlike R-scheme and TwinRoute, it cannot easily handle unexpected sink movements.

Akkaya et al. [17] propose a reactive approach to data forwarding to mobile sinks. They explore the tradeoff between the need for frequent re-routing to handle sink mobility and the communication overhead due to re-routing. Unlike P-scheme and TwinRoute, their approach does not exploit inherent patterns of sink mobility. The AROT approach [18] reduces the amount re-routing of a tree-based approach by only updating routes that have been made redundant by recent sink movement.

Gandham et al. [3] assume control over the locations of mobile sinks, and use an integer linear program to periodically determine the new sink locations. Luo et al. [4] suggest that the best mobility strategy is to follow the periphery of a circular deployment area. Wang et al. [5] aim at optimising network lifetime by jointly determining the movement of the sink and the sojourn time at different points in the network. Luo et al. [2] assume that the mobile sink visits a fixed sequence of anchor nodes, they adjust the pause times at each anchor, and propose MobiRoute, a multi-hop routing protocol that deals with sink mobility. Unlike these papers, our work does not assume fixed or controlled sink movement.

Delay-tolerant Data Collection is typically used in applications like habitat monitoring, where the delivery of data to the sinks is not time critical. A permanent connection between data source and sink is, therefore, not necessary. We divide this work into two sub-classes depending on whether single-hop or multi-hop communication is used for data delivery:

Single-hop communication. Jain et al. [19] physically move MULEs (Mobile Ubiquitous LAN Extensions) to transport data from sensor nodes to sink nodes. The MULE approach utilises single hop communication, thereby generally avoiding the need of routing at the expense of latency. Randomly moving MULEs can potentially take an infinite amount of time to reach a sink, thereby not being able to guarantee an explicit delay threshold of individual data items.

Instead of relying on random movement, [20], [21] rely on controlling the mobile sink to guarantee a data delivery threshold. Somasundara et al. [21] provide practical scheduling algorithms that enable mobile sinks to visit busy sensor nodes more frequently, so that there are no data lost due to buffer overflow. The PBS (Partitioning Based Scheduling) algorithm presented by Ekici et al. [20] groups nodes together that have similar buffer overflow times, and computes mobile sink trajectories based on knowledge of the data generation rate of sensors and their locations. Chakrabarti et al. [22] assume a fixed mobile sink trajectory, and analyse how the transmission range and data rates of sensor nodes impact the power consumed in transferring data to the mobile sink.

For data collection protocols that utilise single-hop communication, a large data buffer is required to enable storage of all generated data between mobile sink visits. Additionally, visiting all nodes individually can often be infeasible due to difficult terrain, and also logistics problems or danger if in a hostile environment. Our proposed algorithm, TwinRoute, avoids these problems by combining mobile sink movement with multi-hop communication among static sensors.

Multi-hop communication. TwinRoute makes similar assumptions to this final class of algorithms, i.e. multi-hop communication and delay-tolerant collection of sensor data. The MRME algorithm [20] extends the PBS algorithm by additionally relaying urgent messages using multi-hop routing to sensor nodes that guarantee data delivery within a delay threshold. Similar to MRME and P-scheme, Chen et al. [6] estimate the interarrival time of an uncontrolled mobile sink at sensor nodes to decide on whether to forward data in a multi-hop manor or store data locally for timely data collection. Like the P-scheme, Chen et al. and MRME exploit inherent patterns of sink mobility, and cannot easily handle unexpected sink movements. In SCAR [23], nodes determine whether a neighbour is a suitable carrier for their data based on its collocation with the sinks, its change in degree of connectivity and its battery level. It is similar to the P-scheme in that nodes that are frequently visited by mobile sinks tend to attract and forward data on behalf of nearby, less visited, nodes.

Somasundara et al. [1] present another delay-tolerant approach that utilises multi-hop routing. They assume a fixed sink trajectory and establish data dissemination trees rooted at nodes within reach of the trajectory. The mobile sink visits for longer periods those nodes that gather data from larger subtrees. Kansal et al. [24] propose adaptive algorithms to control mobility, and propose modifications to directed diffusion to allow for efficient data routing from multiple static nodes to a single mobile sink. MobiQuery [7] is a reactive data collection algorithm that predicts the movement of a mobile sink in order to wake sleeping nodes in the predicted area, thus enabling the nodes to respond to sink queries. Unlike R-scheme, the queries posed in MobiQuery are of spatio-temporal nature in that a data interest exists only for nodes

[2]Predicting the mobile sink trajectory can be actively conducted in an energy efficient way as described in [15], [16]

within a fixed distance of the sink.

To our knowledge, TwinRoute is the first delay-tolerant protocol that combines proactive and reactive multi-hop routing, does not rely on controlled sink mobility, and is suitable for scenarios that involve both predictable and random sink movements.

VI. Conclusions and Future Work

This paper includes a study of three classes of data collection algorithms - variants of existing proactive and reactive schemes, and variants of a novel hybrid scheme called TwinRoute. Our findings are as follows: 1) the performance of pure and hybrid schemes significantly depends on their parameter settings; all schemes trade communication cost for delivery ratio of fresh packets; an increase in the storage delay threshold parameter, decreases the communication cost of P-scheme and TwinRoute, at the expense of lower delivery ratio; an increase in the tree depth parameter, increases the delivery ratio of R-scheme and TwinRoute at the expense of higher communication cost. 2) Given application requirements for data freshness, a network designer can choose the most energy-efficient variant of P-scheme, R-scheme and TwinRoute; in applications where sinks move in a very random manner, the best R-scheme typically outperforms the best P-scheme; in applications where sink traces are more predictable, the best P-scheme often outperforms the R-scheme; the proposed TwinRoute protocol outperforms the pure schemes in almost all cases. 3) The application scenario where TwinRoute features the highest communication savings with respect to P-scheme and R-scheme is one that involves a mix of time-critical and delay-tolerant packets, requires the vast majority of packets to be delivered within their freshness thresholds, and involves both highly predictable and unpredictable sink movement. TwinRoute also featured significant communication savings when tested in an animal monitoring scenario using real zoologist traces and a realistic sensor deployment for monitoring badgers.

In the future, we will analyse the algorithms using an extended set of WSN scenarios that also vary in their sensor topology and data generation rate. Furthermore, we plan to design adaptive pure and hybrid collection algorithms that adjust their parameters by leveraging feedback from sinks about the delay of collected packets. We are particularly interested in designing an adaptive version of TwinRoute that is capable of changing the balance of its proactive and reactive elements in response to changing sink mobility and network conditions.

References

[1] A. Somasundara, A. Kansal, D. Jea, D. Estrin, and M. Srivastava, "Controllably mobile infrastructure for low energy embedded networks," *IEEE Transactions on Mobile Computing*, vol. 5, no. 8, 2006.

[2] J. Luo, J. Panchard, M. Piórkowski, M. Grossglauser, and J.-P. Hubaux, "Mobiroute: Routing towards a mobile sink for improving lifetime in sensor networks," in *DCOSS*, 2006, pp. 480–497.

[3] S. Gandham, M. Dawande, R. Prakash, and S. Venkatesan, "Energy efficient schemes for wireless sensor networks with multiple mobile base stations," in *Global Telecommunications Conference(GlobeCom), Vol. 1*, 2003, pp. 377–381.

[4] J. Luo and J.-P. Hubaux, "Joint mobility and routing for lifetime elongation in wireless sensor networks," in *Intl Conference on Computer Communications (Infocom)*, 2005, pp. 1735–1746.

[5] Z. Wang, S. Basagni, E. Melachrinoudis, and C. Petrioli, "Exploiting sink mobility for maximizing sensor networks lifetime," in *Annual Hawaii Intl Conference on System Sciences*, 2005.

[6] C. Chen, J. Ma, and K. Yu, ""designing energy efficient wireless sensor networks with mobile sinks"," in *"Proc. of ACM Sensys'06 Workshop WSW"*, 2006.

[7] C. Lu, G. Xing, O. Chipara, C.-L. Fok, and S. Bhattacharya, "A spatiotemporal query service for mobile users in sensor networks," *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on*, pp. 381–390, June 2005.

[8] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: accurate and scalable simulation of entire tinyos applications," in *1st Intl Conference on Embedded networked sensor systems (Sensys)*, 2003, pp. 126–137.

[9] H. Luo, F. Ye, J. Cheng, S. Lu, and L. Zhang, "TTDD: Two-tier Data Dissemination in large-scale wireless sensor networks," *ACM/Kluwer Mobile Networks and Applications (MONET), Special Issue on ACM MOBICOM*, 2003.

[10] C.-J. Lin, P.-L. Chou, and C.-F. Chou, "Hcdd: hierarchical cluster-based data dissemination in wireless sensor networks with mobile sink," in *Intl Conference on Wireless communications and mobile computing (IWCMC)*, 2006, pp. 1189–1194.

[11] H. Kim, T. Abdelzaher, and W. Kwon, "Minimum-energy asynchronous dissemination to mobile sinks in wireless sensor networks," in *1st Intl Conference on Embedded networked sensor systems (Sensys)*, 2003, pp. 193–204.

[12] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 2–16, 2003.

[13] J. Jetcheva and D. Johnson, "Adaptive demand-driven multicast routing in multi-hop wireless ad hoc networks," in *ACM Intl Symposium on Mobile ad hoc networking & computing (MobiHoc)*, 2001, pp. 33–44.

[14] P. Baruah, R. Urgaonkar, and B. Krishnamachari, "Learning-enforced time domain routing to mobile sinks in wireless sensor fields," in *IEEE Intl Conference on Local Computer Networks (LCN)*, 2004.

[15] S. Ray, D. Starobinski, A. Trachtenberg, and R. Ungrangsi, "Robust location detection with sensor networks," in *IEEE JSAC (Special Issue on Fundamental Performance Limits of Wireless Sensor Networks)*, 2003.

[16] J. Albowicz, A. Chen, and L. Zhang, "Recursive position estimation in sensor networks," in *Intl Conference on Network Protocols (ICNP)*, 2001, pp. 35–41.

[17] K. Akkaya and M. Younis, "Energy-aware routing to a mobile gateway in wireless sensor networks," in *Global Telecommunications Conference Workshops (GlobeCom)*, 2004, pp. 16–21.

[18] K. I. Hwang, T. Y. Kim, and D. S. Eom, "Proactive data delivery scheme with optimal path for dynamic sensor networks," in *UIC*, 2007, pp. 412–421.

[19] S. Jain, R. Shah, W. Brunette, G. Borriello, and S. Roy, "Exploiting mobility for energy efficient data collection in wireless sensor networks," *Mob. Netw. Appl.*, vol. 11, no. 3, pp. 327–339, 2006.

[20] E. Ekici, Y. Gu, and D. Bozdag, "Mobility-based communication in wireless sensor networks," *IEEE Communications Magazine*, vol. 44, no. 7, pp. 56–62, July 2006.

[21] A. Somasundara, A. Ramamoorthy, and M. Srivastava, "Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines," in *IEEE Intl Real-Time Systems Symposium (RTSS)*, 2004, pp. 296–305.

[22] A. Chakrabarti, A. Sabharwal, and B. Aazhang, "Using predictable observer mobility for power efficient design of sensor networks," in *Intl Conference on Information Processing in Sensor Networks (IPSN)*, 2003.

[23] C. Mascolo and M. Musolesi, "Scar: context-aware adaptive routing in delay tolerant mobile sensor networks," in *Intl conference on Wireless communications and mobile computing (IWCMC)*, 2006, pp. 533–538.

[24] A. Kansal, A. Somasundara, D. Jea, M. Srivastava, and D. Estrin, "Intelligent fluid infrastructure for embedded networks," in *Intl conference on Mobile systems, applications and services (MobiSys)*, 2004, pp. 111–124.