An Energy-Efficient Data Dissemination Protocol for Wireless Sensor Networks

by

Dipu Zhang

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.Sc. degree in
Computer Science

School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

Abstract

Fast, reliable and energy-efficient data dissemination is one of the essential features for applications of wireless sensor networks (WSNs). Since the mobile sink was first introduced to the WSNs to prevent the hot spot problem which usually exists in static sink approaches, a number of data dissemination protocols that support sink mobility have been proposed in recent years. In the first part of this thesis, we present a comprehensive state-of-art overview of the major mobile sink protocols. Although many of them have made an effort to minimize the overhead of sink location updates and to maximize the energy efficiency, most of the protocols failed to concern one important fact that sensor energy consumed during the idling state is a major part of the overall energy consumption and dominates the network lifetime. Moreover, the protocols themselves are faced with all kinds of pending issues. Therefore, in the second part of this thesis, we propose E-Trail, a new data dissemination protocol for WSNs with mobile sinks, which combines a simple yet efficient trail generation scheme with a novel sleep-wake mechanism that lets the redundant sensors turn off their radio and sleep for a certain period of time. Experiments are carried out with the network simulator ns-2 to evaluate the performance of the proposed protocol. The simulation results indicate that E-Trail significantly saves sensor energy and improves network lifetime, while its other network performances such as data delay, control overhead and delivery rate are still maintained at a similar or even better level compared to selected protocols.

Acknowledgements

It is my great honour to thank those many people who have helped and supported me in my graduate studies at the University of Ottawa. First, I would like to thank my supervisor, Professor Azzedine Boukerche, for his invaluable teaching and guidance, as well as for his financial support, which is of special significance to me. Second, I would like to thank Professor Thomas T. Tran, Professor Abdulmotaleb El Saddik and Professor F. Richard Yu, for carefully reviewing my thesis and serving as the exam committee. Also, I would like to thank Dr. Richard W. Pazzi at the PARADISE lab, for his generous help and precious advice on my research works.

Finally, I would like to thank my wife and my parents, whose love and encouragement have always sustained me.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The recent development of micro-electronics has made wireless sensor networks (WSNs) a promising technology not only in military operations, such as intrusion detection, target tracking and battlefield surveillance, but also in civilian applications such as health care systems, habitat monitoring, disaster detection and environment monitoring. A wireless sensor network is a network consisting of spatially distributed autonomous sensors which monitor the nearby physical and environmental conditions (*e.g.*, temperature, sound, vibration, pressure, motion, pollutants) [1]. The sensors in a wireless sensor network perform multi-hop data communication to a base station or a network gateway called a sink. In most cases, each sensor is a constrained wireless device with low computation capability, short transmission range, and very limited energy resources stored in a small onboard battery. However, as applications of WSNs usually require sensor nodes be left unattended for a long period for time due to the high maintenance cost or difficulties in accessing the deployment area, energy efficiency becomes a major concern for methods that are used to deliver data within WSNs. The motivation behind this thesis stems from the demand for a proper data dissemination protocol for WSNs which is reliable, energy efficient, and maximizes the network lifetime.

1.1 Motivation

The collected measurements of sensors in WSNs are delivered through many hops to the sink, which then further forwards the information to a data processing center or a human operator. A data dissemination protocol for WSNs determines when and how data is forwarded to the sink within the sensor network. At first, a number of protocols

were proposed with the use of static sinks [9] [17] [20] [21] [34] [41]. In these type of approaches, the static sink must broadcast beacons through the entire sensor area so as to inform each sensor node about its location. The network topology of a typical static sink approach is shown in Figure. 1.1. It can be observed that sensor nodes closer to the static sink (*e.g.*, Sensor A-F) have to function as relays to numerous nodes. Therefore, they usually suffer faster energy depletion, which may lead to a very short network lifetime. This situation is literally called the hot spot problem [4] and it exists in almost every static sink based data dissemination approach. Moreover, because of the network-wide broadcasting, static sink approaches do not scale with network size and may have severe packet collisions.

To prevent the hot spot problem, researchers often seek to exploit the use of a mobile sink, which is constantly changing its position in order to balance the traffic loads and even the energy consumption among sensors. However, the problem then shifts to how to find a valid routing path to the mobile sink and how to efficiently track the mobile sink so that sensor data can be reported continuously. An obvious, but naive solution is to let the mobile sink send out frequent location updates through the entire network just like a static sink, except on a much more frequent basis. It works, however with extremely poor energy efficiency, as a huge part of the sensor energy is consumed by sink location updating rather than data forwarding. Consequently, the sensors will run out of energy soon and the network will quickly die.

In recent years, much research has been published to provide energy efficient data dissemination protocols to mobile sinks [7] [10] [14] [22] [23] [33] [35]. Although the movement of mobile sinks requires regular broadcasting of sink location information to the network, many protocols [46] [47] [48] have adopted certain methods to constrain the broadcasting within a very small local area while the network connectivity is still maintained at an acceptable level. For example, the Dual-Sink protocol [48] proposes the use of a mobile sink combined with a static sink. The mobile sink helps to alleviate the hot spot problem; however, instead of network-wide broadcasting, the mobile sink only broadcasts location information to a neighborhood of limited hops once in a while. For source nodes that have routes to both the static sink and the mobile sink, they will determine their destination sink based on the forwarding cost; for those with a lack of valid routes to the mobile sink, they will send data to the static sink directly. The Local-based Update Routing Protocol (LURP) [46] introduces the concept of a destination area and limits the location updates within the destination area most of the time. The mobile sink first initiates a global broadcasting to build a round local destination area; then,

as long as it is moving within the destination area, the sink only needs to broadcast its location information to sensor nodes that are located in the area. When the mobile sink eventually moves out of the current destination area, it initiates another round of global broadcasting and builds a new destination area.

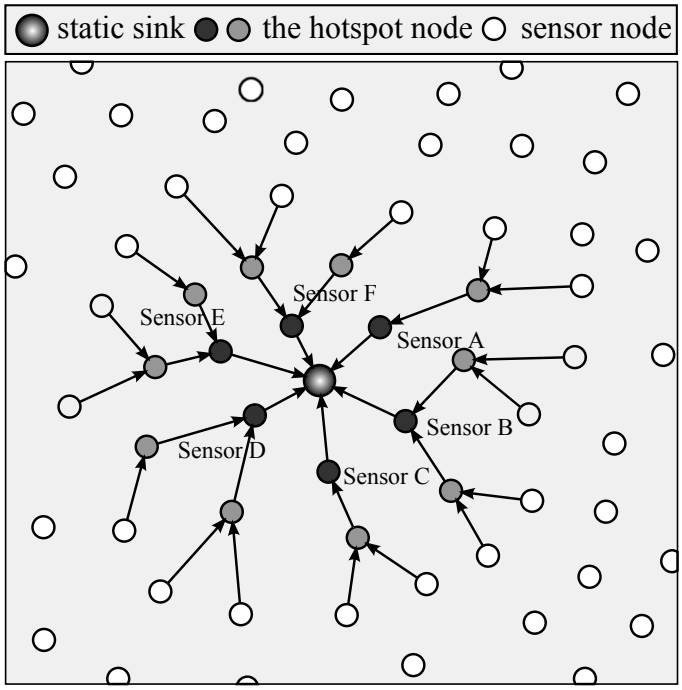


Figure 1.1: Network topology of static sink approaches

Although these protocols have proposed various methods to exploit the energy efficiency of data dissemination process with mobile sinks in WSNs, they failed to consider one important fact that the energy consumed when nodes fall into an idling state dominates the overall energy consumption. According to the research of M.Stemm *et al* [43] and Y.Xu *et al* [49], the energy consumption for $idle : receive : send$ ratios in WSNs are at least $1 : 1.05 : 1.4$, respectively. In fact, according to our investigations and simulation results, the proportion of idle energy consumption can be much higher depending on different network configurations and data sending intervals. Therefore, to further improve the energy efficiency and prolong the network lifetime, new protocols should be developed to find a way to turn off the radio of the "unnecessary nodes" and still maintain valid data dissemination paths to the mobile sink. Doing this will not only reduce the transmission overhead of non-specific packet forwarding processes, such as location

broadcasting, but also conserve sensor energy both in overhearing neighbor data transfer and in idling state energy dissipation when no traffic exists.

1.2 Contribution

The major contribution of this thesis is to design and implement a proper data dissemination protocol for WSNs, which can support multiple mobile sinks and provide accurate, reliable and much more energy efficient data delivery within the sensor network. Towards this objective, this thesis addresses the following:

- A thorough study of major data dissemination protocols for WSNs with mobile sinks is presented. The study investigates the classification and characteristics of mobile sink protocols, describes the concept and algorithm of each protocol, and compares them according to different criteria.

- A novel protocol called "Energy-Efficient Trail-based Data Dissemination Protocol for Wireless Sensor Networks with Mobile Sinks (E-Trail)," is proposed. E-Trail combines a trail-based approach that relies on the concept of a marked or beaten path left by a moving human body as through the woods, with a sleep-wake mechanism that aims to reduce unnecessary energy consumption during the idling state of sensors. The proposed protocol provides accurate, fast and reliable data dissemination to multiple mobile sinks within the sensor network, and its energy efficiency is far better than the other selected protocols, based on our simulation results. The conference paper regarding E-Trail [39] has been accepted by the International Conference of Communications (ICC 2011) and published. The link to download this paper is available at http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5963002.

1.3 Thesis Organization

The rest of this thesis is organized as follows:

- Chapter 2 introduces the related works and presents a literature review of data dissemination protocols for WSNs that support mobile sinks. The general classification and characteristics of mobile sink protocols are discussed in this chapter, and a number of major existing protocols are described and investigated.

- Chapter 3 explains our proposed protocol E-Trail in detail, in terms of the motivation, the important technical concepts, the routing algorithms and the implementations.
- Chapter 4 gives the performance evaluation of the E-Trail protocol and four other selected protocols. The simulation results are presented and analyzed.
- Chapter 5 concludes this thesis by proposing some potential future research to improve the performance of E-Trail.

Chapter 2

Data Dissemination Protocols for WSNs with Mobile Sinks

Researchers of WSNs have noticed that in most static sink approaches, sensor nodes closer to the sink have to constantly relay data for the rest of the network and they unexceptionally suffer fast energy depletion. As long as these sensors run out of energy, the entire network does not function any more, because without them data cannot be delivered to the sink correctly. This situation is literally called the hot spot problem, resulting in a great waste of sensor energy. To avoid the hot spot problem, the use of a mobile sink is proposed, which is continuously changing its position so that the traffic loads and energy consumption can be balanced by its movement. However, previous studies on using mobile sinks either assume that certain global information on the network is available [6] [16] [44] or let the mobile sink convey its location information through frequent network-wide broadcasting [12] [32] [45]. As a result, the network lifetime gains from sink mobility can be offset by the repeated broadcasting which inevitably incurs extra high energy loss. Until recently, several protocols targeted at energy efficiency with mobile sinks are proposed. Although sink mobility requires regular location updates to the network, most of the proposed protocols have adopted certain strategies that successfully constrain the broadcasting among a very limited neighborhood with still acceptable network connectivity. In this chapter, the general classifications, characteristics and features of mobile sink protocols will be presented, followed by the investigation of some major existing protocols. In addition, we compare these protocols in a table based on selected classifications.

2.1 Classification of Data Dissemination Protocols for Mobile Sinks

A data dissemination protocol for WSNs is essentially a combination of several characteristics and features, and the corresponding routing algorithms that accommodate these features. Mobile sink protocols can be classified in many different ways based on their characteristics and features, which may include the application type, the data collection scheme, the disseminated information type, the network architecture and the sink mobility pattern, *etc.*

2.1.1 Application Type

The applications of WSNs generally fall into three major types.

Event-driven In this type of applications, data transmission is driven by designated events [42] [50]. Initially, all sensors are operating in a standby monitoring state and they are programmed to be triggered by certain events. The definition of an event can be chosen arbitrarily and may vary according to different scenarios. For example, the presence of enemy individuals in a military intrusion detection application can be considered as an event. Upon the detection of an event, the responding sensor will propagate the event data (*e.g.*, time, location, decibels of sound, temperature) or a certain form of data announcement messages to the sink, and wait for further directions.

Query-based The sensed data or collected measurements are not forwarded to the sink proactively. Instead, data is first buffered locally and kept within the sensor network. Then, sensors react to data queries from the sink by sending out corresponding measurements and event data. This mechanism is often used in combination with event-driven applications [18] [31]. As the sink is aware of an event that has taken place upon the reception of data announcement messages, it may query the responding sensor for detailed information about the event.

Continuous Data Forwarding Sensors deployed in this type of application are always monitoring environmental conditions such as heat, pressure, light, sound, temperature, *etc.*, and continuously reporting measurements to the sink [15] [46] [47]. Typical continuous data forwarding applications include the weather monitoring systems and urban pollution management systems.

2.1.2 Data Collection Scheme

In static sink approaches, the static sink initially floods the sensor field to build a routing tree with itself as the root and all the sensor nodes as the leaves. Sensors then follow the routing branches to forward data from source nodes to the sink hop by hop. One significant drawback of such design is the network lifetime is severely shortened due to the hot spot problem. Sink mobility helps to eliminate the hot spot problem, and in the mean time, it provides various data collection schemes that can accommodate different sensor features and application requirements.

Multi-hop Communication Similar to static sink approaches, data is forwarded from the source to the mobile sink hop by hop [22] [23] [15] [46] [47]. The multi-hop communication scheme usually has low data delivery delay, which makes it appropriate for time sensitive applications such as traffic control systems and most military related applications. However, as the mobile sink is constantly moving from one site to another, it has to frequently announce its current position to the network. Therefore, without certain constraints, the network-wide broadcasting of sink location information can be overwhelming when the network size becomes too large, and the cost of location updates can be extremely high.

Constrained Multi-hop Communication Instead of flooding the entire network, the mobile sink only sends its location updates to sensor nodes that are less than n hops away and gathers data from those nodes [13] [48]. As a result, the mobile sink collects data from a region of the network at one time, and then moves to another region to continue the process. Energy consumption is significantly reduced this way, since the location broadcasting has been constrained within a limited n-hop area and the average length of the routing paths has been shortened as well. However, data delay cannot be guaranteed as the source nodes in certain regions of the network may have to wait a long time before the mobile sink eventually visits the nearby area and collects data from them. Network delay is often unpredictable in constrained multi-hop communication approaches, especially when the movement path of the mobile sink is not well designed or simply follows a random path. This data collection scheme saves sensor energy and prolongs network lifetime, but at the cost of extended data delay. It is thus only feasible for applications without restricted data freshness requirements.

Direct Source-Sink Communication This data collection scheme is only valid when

there are "powerful nodes" with sufficient energy supplies and long transmission ranges distributed in the sensor network. The powerful nodes form clusters over nearby sensors and act as the cluster heads, which collect measurements and event data locally and transmit data directly to the sink at a high transmission power [26]. One major issue with this direct source-sink communication scheme is it cannot be applied to homogeneous WSNs, where all deployed sensors have the same amount of energy resources and similar transmission ranges. Besides, it also suffers from network vulnerability issues. Once a cluster head is down, the entire cluster will get disconnected from the network.

Passive Data Collection In passive data collection, the mobile sink visits each source one by one and collects data individually from them [11] [13]. As it moves through the sensor field, the sink regularly sends out beacons with TTL=1. Upon the reception of sink beacons, the source nodes that are one hop away from the sink send their buffered data to the mobile sink directly. This scheme provides maximum energy efficiency, however, it also has a considerably longer data delivery delay. Passive data collection is not widely applied in the research studies of mobile sinks, yet it may still be adequate for some delay tolerant applications, such as animal habitat monitoring systems. The movement path of the mobile sink must be carefully designed to make sure that each source node in the sensor network will be visited at least once during a specific period of time.

2.1.3 Disseminated Information Type

Several types of information are disseminated within the wireless sensor network for different purposes. Generally, information exchanged and communicated in the WSNs can be categorized into three major types, based on their functionalities.

Sink Location Sink location information is generated by the sink. It is used to propagate the current position of the mobile sink through the sensor network. It can either be beacons that help sensor nodes to construct and maintain their routing tables [15] [38], or messages that contain geographic coordinates acquired through the use of a GPS system [24] [25] or some virtual coordinate systems [31] [46] [47] [50]. As the mobile sink is constantly moving in the sensor field, sink location information has to be disseminated on a regular basis. The update scope and frequency should be determined by the transmission range of sensors, the current speed of the mobile sink, the pattern of sink mobility and the data collection scheme together.

Sensor Data Sensor data includes sensor readings, collected measurements and event data. Sensor data is delivered to the mobile sink either continuously or upon the reception of sink queries (see the discussion of application type). Some data dissemination protocols may require sensor data to be first sent to a subset of nodes in the network and then collected by the mobile sink (see the discussion of data storage place in the later part of this section).

Meta-data Meta-data has been defined as the information about the sensed data itself. In several approaches, the source node does not attempt to transmit the sensed data immediately. Instead, it stores data in the buffer and only sends a data description message that contains the information about the source or the detected event to the network and waits for further directions. For example, in the TTDD protocol [31], when a source node is triggered by a designated event, it first constructs a virtual grid and sends a data announcement message along the grid. If detailed information about the event is required, the mobile sink sends data queries to the source and the corresponding event data will flow downstream to the sink following the same path of sink queries but in the reverse direction. The data announcement message in TTDD is one form of meta-data, which plays a role in notifying the network and the sink about the event that has just taken place in the sensor field.

2.1.4 Data Storage Place

In most cases, the sensed data is either buffered locally at the sensor node or sent to the mobile sink directly. However, there are several approaches where the sensed data is stored in some other node(s) prior to sink data collection. In this way, certain data pre-processing techniques such as aggregation can be performed to further reduce the overhead of data transmission. Sink mobility has led to great flexibility of which node(s) store the collected measurements and event data within the sensor network.

One Single Node Sensed data of all sources are stored in a particular node which is usually chosen in a deterministic or geographic way. The mobile sink collects network-wide data by visiting only that single node. This approach has extremely low fault tolerance, because once the particular node is not functioning, data from the entire network will be lost. Moreover, it suffers from the hot spot problem as well.

A Subset of Nodes Rather than one single node, a group of nodes are designated to

store the network-wide data. Initially, data from each source is delivered to one node out of this designated group (usually the closest one to the source). Then, the mobile sink is able to collect data by visiting only a small subset of sensor nodes if it has prior knowledge of the distribution of these nodes.

A Set of Nodes for Each Source Node Each source node sends its collected measurements and event data to a set of selected nodes, so that the data is replicated and distributed in different regions of the network. The basic idea is to enable the mobile sink to collect data from the entire sensor network by visiting only a few regions or even one single region of the network. Data redundancy could be a potential problem for this approach.

Only at Source The source stores its own generated data in the buffer. It may either forward the sensed data along valid routing paths to the mobile sink, or only passively wait for the mobile sink to pass by and collect data from it.

2.1.5 Network Routing Architecture

Research of mobile sinks once focused mainly on broadcasting based approaches, also called the flat data dissemination protocols. In flat protocols, the mobile sink has to regularly broadcast its location information through the sensor field so that the routing paths of sensors can be updated. However, as WSNs can become composed of thousands or even tens of thousands of sensors deployed over extremely large terrains, researchers have begun to realize that flat protocols do not scale with large sensor networks. Thus, they often explore ways to organize the underlying physical network into virtual structures such as grids, strips, rings or clusters. Sink queries, location updates and sensor data are then forwarded through these overlay virtual infrastructures. Virtual infrastructure based approaches have often been investigated as more efficient in the presence of mobile sinks when compared to flat data dissemination approaches, especially in very large sensor networks [19].

Flat Data Dissemination Flat data dissemination protocols are feasible for homogeneous WSNs where all deployed sensor nodes participate equally in the query forwarding and data relaying tasks. Sensor modules in this type of protocol are usually not required to be location-aware, and they do not have to maintain a certain form of virtual infrastructure in memory throughout network lifetime. The

mobile sink needs to periodically broadcast its location information to the network to update the routing paths of sensor nodes. Many flat approaches today [15] [46] [47] [48] successfully constrain the location broadcasting within a small neighborhood of sensors. Dynamic repair of the routing paths close to the mobile sink is also proposed [15] [46] [47]. These techniques help to reduce the overhead of sink location updates a great deal. However, as we have mentioned, flat data dissemination protocols do not scale with network size. With thousands or even tens of thousands of sensor nodes deployed in the network, the cost of sink location updates becomes extremely high, and the network throughput can go down even to zero during each round of broadcasting. A lot of data packets can be lost due to the overwhelming network collisions.

Virtual Infrastructure Based Data Dissemination Virtual infrastructure based protocols organize the physical network topology into virtual structures, such as grids [28] [31] [50], strips [18], rings [42] or clusters [30]. These virtual infrastructures usually act as a rendezvous area for storing and retrieving the sensor data or the information about the data source and the event. Sensor nodes that are located in the rendezvous area are designated to forward the sink queries, sink location updates and data. When the mobile sink is moving through the sensor field, the designated sensor nodes nearby are queried by the sink for data. Then, depending on the protocol design, sensor data may either be delivered to the sink directly (when data is stored in the rendezvous area), or forwarded along the virtual infrastructure from the source to a designated node and then further forwarded to the mobile sink (when only the information about the data source and the event are stored in the rendezvous area). For example, in the TTDD protocol [31], each source node that detects an event will build a virtual grid of its own over the physical network. Then, sensor nodes that are closest to the crossing points of the virtual grid become dissemination nodes for that source, which have the information about the source and the detected event as well. Once the mobile sink is aware of the event that has just taken place and requires further event information, it will broadcast a data query within a grid-cell-size local area. The nearby dissemination node that receives the sink query then forwards it along the virtual grid towards the source. Upon the reception of a sink query, the data source sends corresponding event data back to the mobile sink along the query forwarding path, but in an opposite direction.

2.1.6 Sink Mobility Pattern

The sink can follow a variety of mobility patterns to accommodate the different requirements of wireless sensor network applications. The mobile sink may move continuously and randomly in the sensor field while always updating its location to the network. Alternatively, the mobile sink could be following a predetermined path which fully covers the sensor area. Future positions of the mobile sink are therefore predictable to the network sensors, and less routing update messages are needed. The mobility patterns of mobile sinks in WSNs can be categorized as completely random mobility, controlled mobility and predetermined movement path.

Random Mobility Completely random mobility does not assume any predetermined movement path or constrained sink movement. The mobile sink moves randomly in the sensor network to collect sensor data. Frequent sink location updates and route reconstruction are usually necessary to support this pattern. Random mobility provides great scalability to the design and implementation of data dissemination protocols, as the movement of the mobile sink does not assume any network configuration nor rely on any specific application. The problem with random mobility is it may not be optimal in the terms of energy efficiency and network lifetime; however, it is still favored by researchers and is the most widely adopted mobility pattern in mobile sink protocols of WSNs [5] [22] [27] [31] [37] [46] [47].

Controlled Mobility Although there is no predetermined path for the mobile sink with controlled mobility, the sink adjusts its movement direction dynamically to optimize the network lifetime. For example, mobile sinks following this mobility pattern usually have the tendency to visit areas where sensors have more residual energy or sensor data is generated more frequently [7] [8]. The controlled mobility improves the network lifetime in two ways: first, it lowers the overall data forwarding cost and saves sensor energy; second, it balances the residual energy among sensor nodes in the network.

Predetermined Movement Path There are several reasons to adopt a predetermined movement path for the mobile sink: to balance the energy consumption, to maintain the network connectivity, to reduce the sink location update cost and to cover the entire sensor area. For instance, in the Dual-Sink protocol [48], the mobile sink follows a clockwise square-shape movement path which covers most parts of the sensor network. The sink moves along the path, and stops to send out beacons

with very small hop numbers (*e.g.*, $TTL = 2$) when it arrives at a new site. Then, it stays at the current site for a period of time long enough for at least one round of data gathering. After collecting data from the sensors nearby, the mobile sink moves on to another site along the path and repeats the above process. The predetermined sink movement path in the Dual-Sink protocol guarantees that each sensor node in the network will have equal opportunities to receive beacons from the mobile sink. Furthermore, as the mobile sink moves from one site to another along the path in the sensor field, the traffic loads and energy consumption are also balanced. Another example is the PMDD protocol [29], where the data queries sent from the sink contain information on the current speed and moving direction of the mobile sink. With the information provided in the sink queries, the sensor nodes are capable of calculating and predicting the future positions of the sink for the next few seconds. Accordingly, data is forwarded towards the future positions of the mobile sink. The sink movement path of this protocol can be described as to some extent predetermined, at least for the next few seconds. The predetermined movement path helps to lower the frequency of sink location updates and provides better energy efficiency than randomly walking mobile sink approaches. In the most ideal scenario, the sink location updates can be eliminated completely if only all sensors and the mobile sink in the network are time-synchronized and they all have knowledge of the predesigned path for the mobile sink.

2.2 An Overview of Mobile Sink Protocols

We have investigated a number of major existing mobile sink protocols and find it proper to classify these protocols based on their network routing architectures. Two types of data dissemination protocols for mobile sinks will thus be described and investigated in this section: the flat data dissemination protocols and the virtual infrastructure based protocols. In flat data dissemination protocols, the mobile sink has to periodically flood the network in order to announce its location. Sensors reconstruct their routing paths upon the reception of sink location updates. This scheme is very intuitive and easy to implement. Well designed flat protocols often manage to confine the broadcasting within a small sensor area. However, when it comes to very large WSNs, virtual infrastructure based protocols have their advantages. They organize the physical network topology into virtual grids, strips or virtual rings, and convey information through these overlay infrastructures. Data dissemination in the sensor network is thereby performed on top

of virtual infrastructures. The network is then equivalent to a network composed of only those sensor nodes which participate in the construction of the virtual infrastructure, and the number of nodes involved in the data dissemination process is significantly reduced, resulting in much less communication overhead. However, we would like to also point out that virtual infrastructure based protocols do not outperform flat data dissemination protocols in small or mid-size sensor networks as expected, due to the high maintenance cost of the virtual infrastructures and the redundant control overhead of the query-based data collection scheme which is widely applied in virtual infrastructure based approaches.

In this following section, we will present an overview of both types of data dissemination protocols for WSNs with mobile sinks. Eight popular and typical mobile sink protocols with four of each type will be discussed. Essentially, each protocol is merely a combination of different features and characteristics mentioned in the previous section, and corresponding routing algorithms that accommodate these features. A comparison of these eight protocols is listed in Table. 2.2.

2.2.1 The Dual-Sink Protocol

The studies of mobile sink protocols prior to Dual-Sink either assume some global information of the network (*e.g.*, the movement path of the mobile sink) is detectable, or let the sink convey its location information by frequent broadcasting to the entire network. The offset problem was then brought onto the table: the network lifetime gain of WSNs by sink mobility may well be counteracted by the high energy loss during repeated network-wide broadcasting. To address this problem, X. Wu et al. propose to use both a static sink and a mobile sink in the sensor field (see Figure 2.1). The static sink broadcasts its location through the entire sensor field for just one time at the very beginning. As for the mobile sink, it broadcasts beacons only to a nearby subset of sensor nodes in the network when it arrives at a new site on the predetermined square-shape movement path. Consequently, each node in the wireless sensor network of Dual-Sink will have at least one routing entry to the static sink. For those nodes that are linked to both sinks, they can send their data to the nearer one.

The data dissemination process of the Dual-Sink protocol involves three types of devices, namely, the static sink, the mobile sink and the sensor nodes. The static sink broadcasts a HELLO message to the entire network only one time, when the network starts to operate. The information propagated in the HELLO message contains the type of the sink, the intermediate node that forwards the message and the hop distance. The

hop distance is equal to the number of hops it takes for the HELLO message to reach the sensor node from the sink. After this one round of broadcasting, each sensor in the network is linked to the static sink, and the static sink is ready for receiving data.

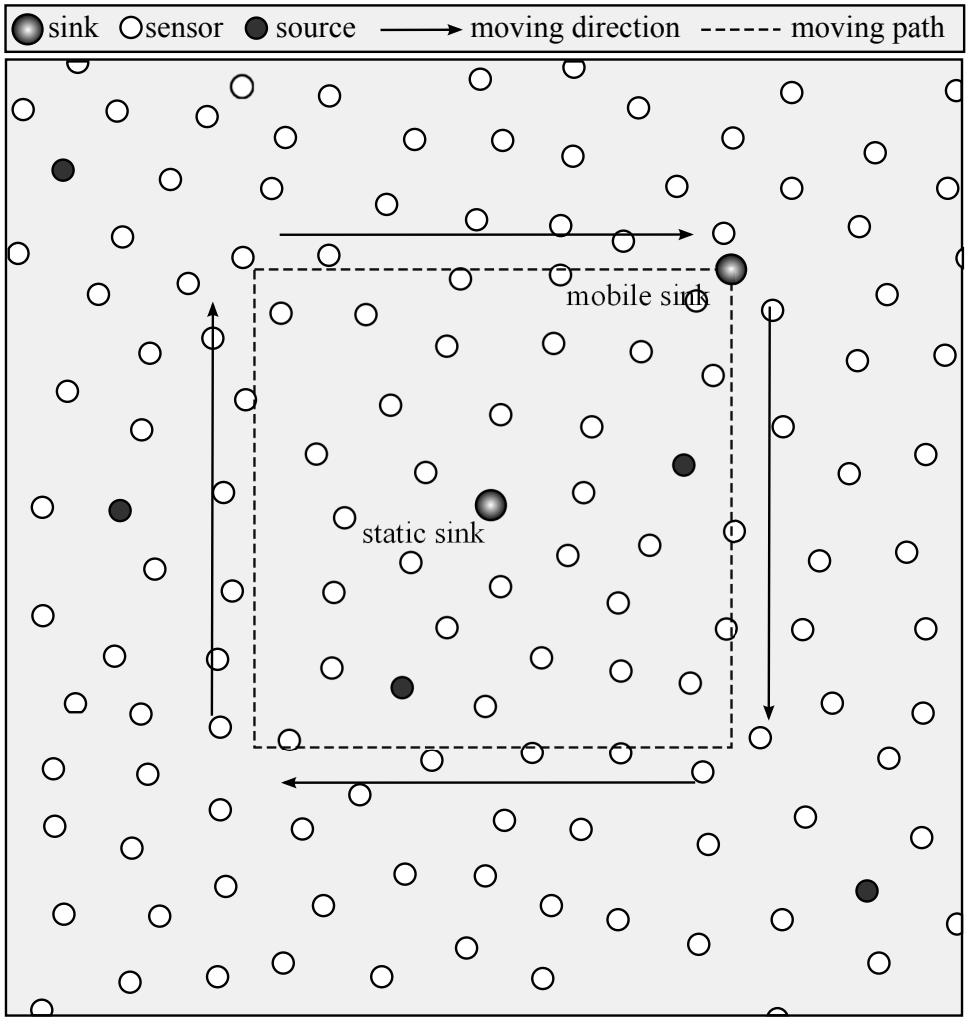


Figure 2.1: Network topology of the Dual-Sink protocol

For the mobile sink, it also broadcasts a HELLO message every time it arrives at a new site along the path. The only difference between the static sink HELLO messages and the mobile sink HELLO messages is the mobile sink HELLO messages are constrained by a limited TTL (the Time-To-Live parameter of the network packets) and can only be forwarded to a small subset of sensor nodes in the network. The authors of the Dual-Sink protocol assume the mobile sink will stay at the current site long enough to gather data

from the nearby sensors after the broadcasting of HELLO messages, and then move on to another site. They also assume all nodes generate data at a rather low frequency so that no data will be generated and transmitted before the mobile sink arrives at the next site. However, we doubt these are very realistic or practical assumptions; they may in fact limit the applications of this protocol.

SinkType	**HopDistance**	**NextHopID**
StaticSink	9	C
StaticSink	9	D
MobileSink	3	B
MobileSink	3	E

Table 2.1: An example of the routing tables of Dual-Sink sensor nodes

Sensor nodes deployed in the network store the routing entries both to the static sink and to the mobile sink in their routing tables (see Table. 2.1 for an example). When the sensor node receives a HELLO message, it checks the type of the sink and uses the hop distance to refine and update its routing table corresponding to the specific sink. The pseudo code for the processing of a HELLO message is presented in Figure. 2.2. When a DATA message is generated or received, the sensor node selects the next hop for data delivery from its routing table. The nearer sink with a smaller hop distance in the table is chosen as the destination sink of this DATA message, and the corresponding intermediate node is chosen as the next hop. If multiple routing entries are associated with the same sink in the table, the intermediate node with the most residual energy will be selected. For those nodes that have not received any HELLO message from the mobile sink or when the mobile sink routing entries are not valid any more, the static sink is chosen automatically since the value of the hop distance for the mobile sink is set to INFINITY when the network starts or if the routing path to the mobile sink expires, *i.e.*, the mobile sink has left the nearby area.

Dual-Sink is the first protocol that addresses the offset problem caused by repeated network-wide broadcasting from the mobile sink. It is a distributed flat data dissemination protocol which aims to alleviate the hot spot problem in static sink approaches. Dual-Sink provides energy efficiency to the WSNs with a combination of the static sink and the mobile sink. However, the potential application domain of this protocol is very limited because it has made a number of unrealistic assumptions in the design. In addition, we argue that the effect of the mobile sink balancing the traffic loads and energy

```
On receiving a HELLO message
// static sink HELLO
IF hello.sinkType == SS THEN
    IF (hello.hopCount + 1) > ss.hopDis THEN
        discard message hello;
        return;
    ENDIF

    IF (hello.hopCount + 1) < ss.hopDis THEN
        // update the current hop distance
        ss.hopDis = hello.hopCount + 1;
        clear rTable entries associated with static sink;
    ENDIF

    IF (hello.hopCount + 1) <= ss.hopDis THEN
        append new SS entry to rTable;
        IF hello.TTL - 1 == 0 THEN
            discard message hello;
        ELSE
            hello.hopCount = hello.hopCount + 1;
            hello.TTL = hello.TTL -1;
            forward message hello;
        ENDIFELSE
        return;
    ENDIF
ENDIF
// mobile sink HELLO
IF hello.sinkType == MS THEN
    follow the same process as described above only to
                routing entries associated with mobile sink;
ENDIF
```

Figure 2.2: Pseudo code for the processing of HELLO messages

consumption among sensors can weaken as the network size becomes larger. For instance, in a sensor network with approximately a thousand nodes deployed, if the mobile sink only broadcasts beacons to sensor nodes less than two or three hops away as the protocol suggested, it hardly has any positive influence to the rest of the network. Therefore, the Dual-Sink protocol will probably degrade to a static sink approach in larger sensor networks, faced with the threat of the hot spot problem again.

2.2.2 The Efficient Routing Protocol

The Efficient Routing protocol proposed by Fodor et al. [15] is the first mobile sink protocol for WSNs that supports multiple mobile sinks with no restriction on sink movement patterns and has no requirement for the subscription of data in advance. It is a flat data dissemination protocol which uses restricted broadcasting to update the location of mobile sinks in the network. The protocol tries to find a compromise between the optimal routing paths to the mobile sinks and the number of messages needed to update these routes. The authors claim that since nodes farther away from the sink send their data through many hops, a slightly higher forwarding cost along the routing path does not impose a major performance degradation compared to the energy gain from using much fewer route updates. Therefore, the only routes that should be updated are those which forward sensor data in a totally wrong direction and those which are far longer than the optimal routing paths. In other words, nodes farther away from the sink should be updated less regularly than nodes closer to the sink.

Each node in Efficient Routing maintains a routing table containing a list of neighboring nodes that can be used as the intermediate nodes to forward data to mobile sinks. When a node receives a data message, it selects the nearest mobile sink with the smallest registered cost (a concept similar to the hop distance in Dual-Sink) as its destination sink, and the corresponding intermediate node as its next hop. The major contribution of the Efficient Routing protocol is a unique method to process sink location updates received by sensors. It employs a different update mechanism from Dual-Sink so that the sensor nodes farther away from the sink will be updated less regularly than the sensor nodes closer to the sink. The concept of cost is introduced, which can be chosen arbitrarily, such as the number of hops from the mobile sink or some QoS related metrics. Two types of costs are used in this protocol: the registered cost at each sensor node, which represents the "registered" (may not always be valid though) forwarding cost from this node to a specific mobile sink, and the actual cost, which is propagated in sink location

update messages. A threshold is then defined related to the ratio of the registered cost and the actual cost. Upon the reception of a location update message from the sink, the sensor node compares the actual cost in the message and its own registered cost. If the change between the two costs exceeds a certain limit (*i.e.*, the threshold), the routing table and the registered cost of this node are updated, and the message is further forwarded to other nodes. Otherwise, the update message is simply discarded. Figure. 2.3 illustrates the algorithm for sensor nodes to process the sink location updates in the Efficient Routing protocol.

Since the sensors do not distinguish different rounds of sink location updates, routing branches may split off from the routing tree easily due to the special update mechanism. For example, as illustrated in Figure. 2.4, initially the mobile sink is within the transmission range of Node A, so all other nodes will transmit their data to the sink through Node A. The registered costs of Node A, B, C, D, E, F, G are respectively 1, 2, 3, 4, 5, 6, 7. After a time slot t, the mobile sink moves into the transmission range of Node G but out of the range of the previous Node A. If the sink sends out location updates at this time, only less than half of the nodes (Node E, F, G) can be updated. This is because when the update message reaches Node D, the actual cost propagated in the message is 4. However, the registered cost of Node D is also 4. Therefore, no matter what the threshold is, the routing information of Node D will not be updated in any way, because there is no difference between the actual cost and the registered cost. As a result, the routing branch from Node D to Node A splits off from the routing tree and gets disconnected from the mobile sink. To resolve this problem, the Efficient Routing protocol proposes to use an expanding ring search method for path recovery. As long as the immediate ex-neighbor of the sink (Node A in our example) detects the disappearance of the mobile sink (this can be done by setting a timer, and if the node has not received any newer update from the sink before the time expires, the sink will be considered to have moved away), it will broadcast a discovery message to its one-hop neighbors. Nearby sensor nodes will respond to this discovery message if they have newer information about the sink. If no one returns a message, the immediate ex-neighbor of the sink will increase the TTL and broadcast the discovery message again until some sensor nodes respond to it or the TTL reaches a limit.

The Efficient Routing protocol is a simple yet efficient approach that reduces the overhead of sink location updates to a great deal. The average data delivery delay may increase a little bit, as the routing paths are usually not optimal because of the update mechanism used by sensor nodes in this protocol. However, compared to the

```
On receiving a sink location update message msg

// if msg is sent directly from a sink or if the node
// has no route to the specific sink
IF msg.cost == 1 or registeredCost[msg.sinkID] == INFINITY THEN
    // clear the routing entries related to the specific sink
    clear rTable[msg.sinkID];
    registeredCost[msg.sinkID] = msg.cost;
    updateSeqNo[msg.sinkID] = msg.seqNo;
    add new routing entry to rTable;
    msg.cost = msg.cost + 1;
    broadcast location update msg;
    return;

ELSE IF registeredCost[msg.sinkID] != INFINITY
        and updateSeqNo[msg.sinkID] < msg.seqNo THEN
    // calculate the ratio of cost change
    costChangeRatio = |registeredCost[msg.sinkID] - msg.cost|
                        / registeredCost[msg.sinkID];
    // if the cost change is above a predefined threshold
    IF costChangeRatio > THRESHOLD THEN
        clear rTable[msg.sinkID];
        registeredCost[msg.sinkID] = msg.cost;
        updateSeqNo[msg.sinkID] = msg.seqNo;
        add new routing entry to rTable;
        msg.cost = msg.cost + 1;
        broadcast location update msg;
        return;
    ELSE
        drop message msg;
        return;
    ENDIFELSE
ENDIFELSE
```

Figure 2.3: Pseudo code for the processing of sink location updates

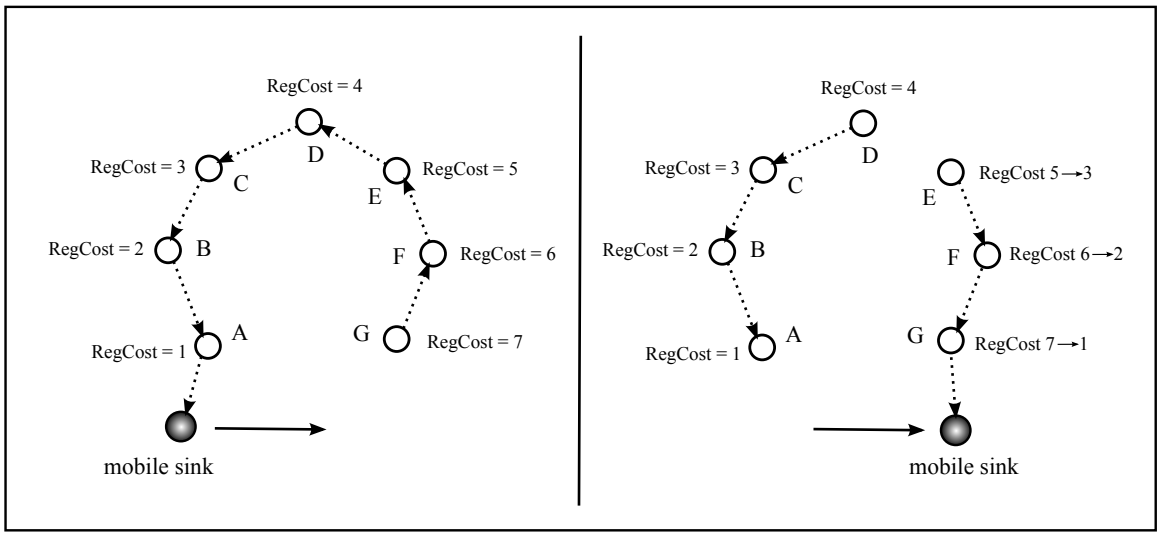


Figure 2.4: Routing branch splits off from the routing tree

great energy gain from using much less location updates, the extended data delay is worthy and acceptable. One critical parameter of this protocol is the threshold, which is related to the ratio of the registered cost and the actual cost. The threshold must be carefully set and tested since its value may well affect the performance of this protocol. The simulation result from the original paper suggests the proper value of the threshold is around 40% to 60%, at which the energy gain and the increase in average route length are well balanced. Nevertheless, it may not apply to every case and still needs to be investigated.

2.2.3 The Local Update-Based Routing Protocol

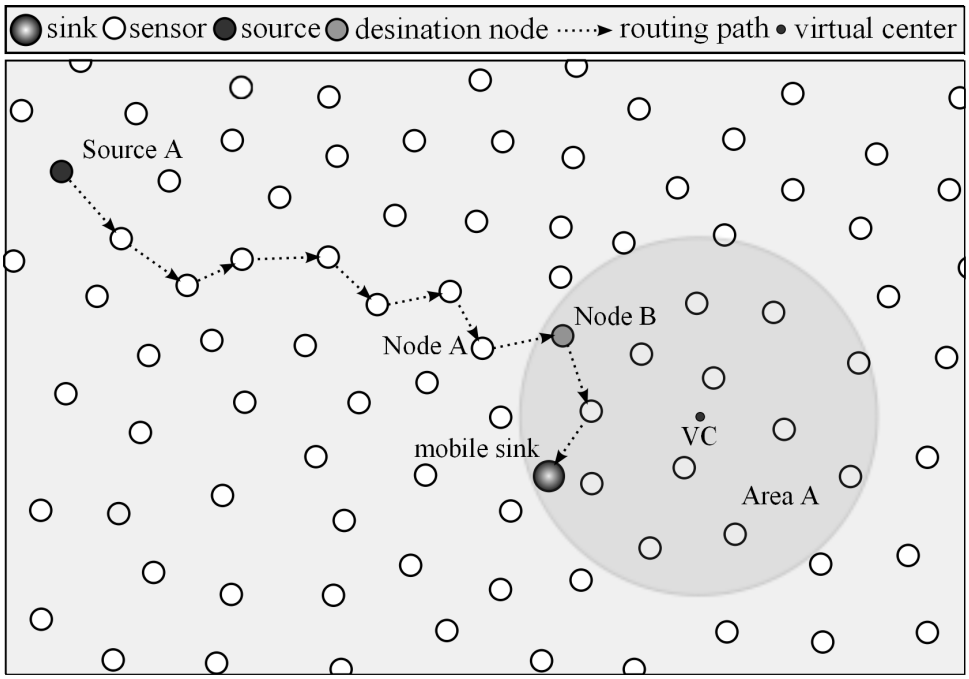
We discussed in Chapter 1 that a very intuitive and easy strategy to support mobile sinks in the WSNs is to let the mobile sink frequently broadcast its location information through the entire network. The strategy is feasible yet not energy efficient, because network-wide broadcasting can be overwhelming when the network size becomes too large. The Local Update-based Routing Protocol (LURP) proposed by Wang et al. [46] suggests a new mechanism to constrain the location broadcasting within a local area rather than through the entire sensor field. In LURP, as long as the mobile sink is moving within a small round area called the destination area, it only broadcasts location updates to the sensors inside the destination area. When the sink moves out of the

current destination area, a new destination area will be built through an extra round of network-wide broadcasting, and the same local update process will be repeated within the new destination area. LURP is a flat data dissemination protocol. Unlike other flat protocols discussed previously in this chapter, LURP assumes sensor nodes in the network to be location-aware, *i.e.*, each node knows its own location, its one-hop neighbors and the location of the mobile sink through certain localization algorithms.

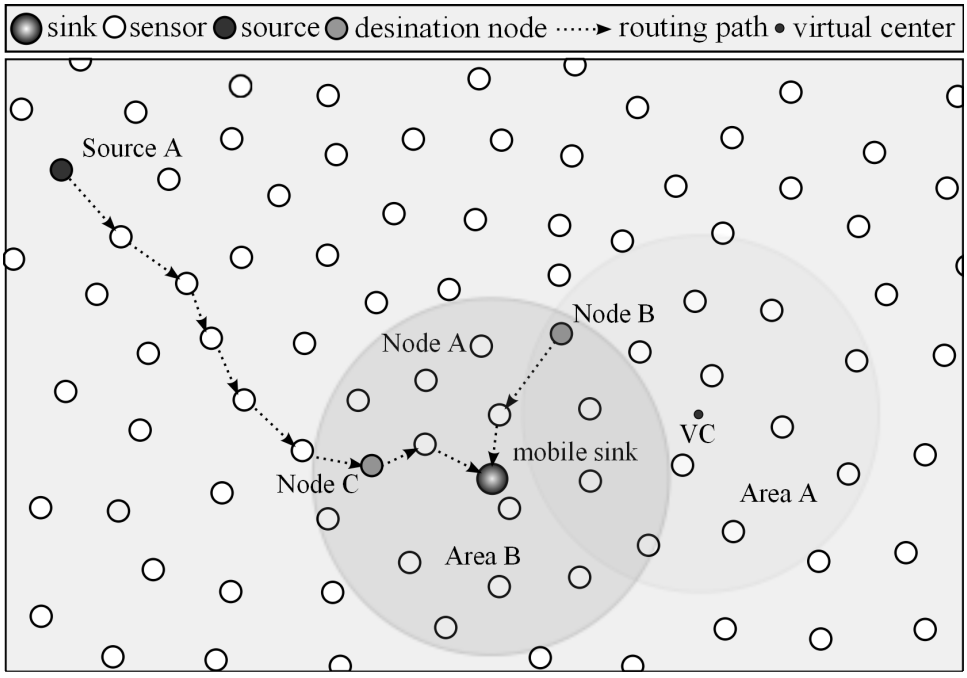
Figure. 2.5 illustrates the data dissemination of the LURP protocol. When the network starts to operate, the mobile sink initially broadcasts a beacon through the entire network so that each sensor will have a valid route to the sink. A virtual round destination area is built centered at the current position of the mobile sink, which is called the Virtual Center (VC), see Figure. 2.5(a). Then, as long as the mobile sink is moving inside the destination area (Area A), it only needs to update its location to the nodes located in this area. Consequently, data dissemination of LURP from the source to the mobile sink involves two stages of routing processes. In the first stage, sensor data is forwarded from the source to the current destination area, or, more specifically, it is forwarded to a sensor node called the dissemination node at the edge of the destination area. For example, in Figure. 2.5(a) Node B is the dissemination node of destination area Area A for Source A, as it was originally selected as the next hop of Node A for data delivery according to the initial network-wide broadcasting. When a data packet reaches the destination area, the second stage of the routing processes begins and further forwards this data packet to the mobile sink following a shortest path.

Once the mobile sink moves out of current destination area (see Figure. 2.5(b)), it will have to broadcast its location information to the entire network again to rebuild a new destination area (Area B). After that, the data dissemination process simply repeats itself, in the way described above, within this new destination area. In case some sensor nodes far away from the mobile sink have not received the updated sink location information and are unaware of the expiration of the old destination area, they can still send their data to the old dissemination node (Node B). The old dissemination node will forward the data to the new destination area anyway because it has been updated about the new location of the mobile sink through the last round of network-wide broadcasting.

As the mobile sink is location-aware, it can detect whether it is out of the current destination area by regularly calculating the distance between its current position and the Virtual Center (VC). If the distance is shorter than the radius of the destination area, the sink sends out local updates only; otherwise, the sink initializes a new round of network-wide broadcasting (Figure. 2.6). Similarly, the sensor node can decide if it



(a)



(b)

Figure 2.5: Data dissemination process of the LURP protocol

belongs to the current destination area by calculating the distance between the VC and its own position. On receiving a local update message from the sink, the sensor belonging to the destination area will update its routing table and forward the update message to other nodes, or, it will discard the message (Figure. 2.7).

```
// initially when the mobile sink starts
broadcast global location updates;
construct a destination area centered at itself;
start checkPostionTimer;
WHILE the mobile sink is moving within the sensor field
    IF checkPostionTimer expires THEN
        distanceToVC = distance(currentPos, VCPos);
        // if the sink moves out of the destination area
        IF distanceToVC > RADIUS THEN
            broadcast global location updates;
            construct a new destination area;
            reset checkPostionTimer;
            // if the sink is still in the destination area
        ELSE
            broadcast local updates to the destination area;
            reset checkPostionTimer;
        ENDIFELSE
    ENDIF
ENDWHILE
```

Figure 2.6: Pseudo code of the mobile sink in LURP

LURP is a protocol that manages to constrain the frequent location updates of the mobile sink in a small local area for most of the time during network operation. Compared to those protocols where the mobile sink frequently updates its location to the entire network, LURP dramatically reduces the control overhead and network collisions. Meanwhile, the movement of the mobile sink is continuously tracked by the nearby sensors in the destination area, and the network connectivity is still maintained at a high level. The routing paths from the source to the dissemination node and from the dis-

```
On receiving a location update message msg
// if a sensor node out of the destination area
// receives a local update message
IF msg.beaconType == LOCAL_UPDATE and
          distance(nodePos, msg.VCPos) > RADIUS THEN
    drop msg;
    return;
ELSE
    // local and global updates share a same seqNo
    IF msg.seqNo > updateSeqNo THEN
        updateSeqNo = msg.seqNo;
        clear rTable;
        hopDis = INFINITY;
    ENDIF
    IF msg.hopCount > hopDis THEN
        drop msg;
        return;
    ENDIF
    IF msg.hopCount < hopDis THEN
        clear rTable;
    ENDIF
    hopDis = msg.hopCount;
    add new entry to rTable;
    msg.hopCount = msg.hopCount + 1;
    broadcast location update msg;
    return;
ENDIFELSE
```

Figure 2.7: Pseudo code for the processing of location update messages

semination node to the mobile sink in the destination area follow the shortest routing paths separately, but the overall data forwarding path may not always be the optimal one. However, this does not necessarily impose a great extension on data delay. The size of the destination area in LURP is very important, since it can affect the frequency of network-wide broadcasting, which is key in reducing the overhead and sensor energy consumption. If the destination area is too small, the mobile sink will have to keep switching from one destination area to another and flood the entire sensor network quite often. If the destination area is too large, the local update cost within the area will increase and have a chance to offset the energy gain from using less flooding. Unfortunately, the protocol failed to give a clear method to calculate the proper size of the destination area. From our point of view, it should be determined by considering the network size, the radio transmission range of sensor nodes, and the moving speed of the mobile sink together.

2.2.4 The Adaptive Location Update-Based Routing Protocol

The Adaptive Location Update-based Routing Protocol (ALURP) [47] is an ameliorated version of the LURP protocol. Based on a similar local update mechanism to LURP, the mobile sink in ALURP adaptively adjusts its local broadcasting range while it is moving inside the destination area. This minor improvement further reduces the overhead of sink location updates and the sensor energy consumption.

In the LURP protocol, once the radius of the destination area has been set, the mobile sink will have to broadcast its location to all the sensor nodes inside the current destination area every time. However, this may not be necessary under some circumstances. The ALURP protocol claims that the local update range of the mobile sink can be further restricted into a smaller area which is also centered at the virtual center (VC), but with a radius of the distance between VC and the current position of the sink. For example, in Figure. 2.8, if the mobile sink built a destination area (Area A) centered at VC (Pos A) and then it moves to another position (Pos B), the sink may only need to update its location to sensors in Area B, which is called the adaptive area in ALURP. The radius of this area changes as the sink moves. When the sink initially builds a new destination area, the radius of the adaptive area is considered as 0. Then, as the sink moves farther away from VC, the radius of the adaptive area increases accordingly and the sink only broadcasts to the sensor nodes located in the current adaptive area. Similar to its predecessor LURP, if the mobile sink eventually moves out of the destination area,

it starts another round of network-wide broadcasting and builds a new destination area.

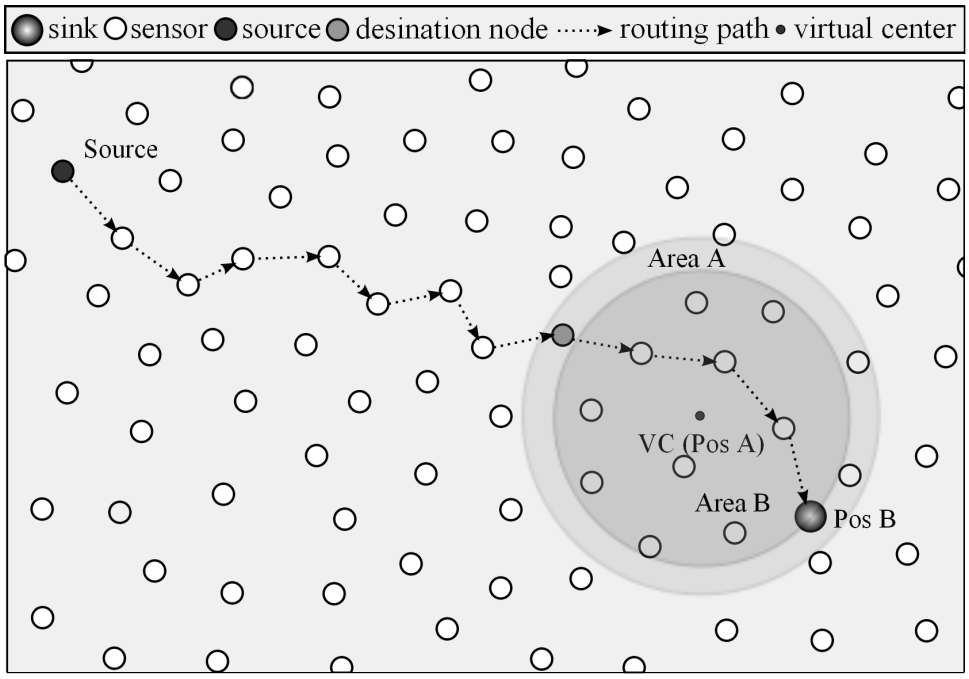


Figure 2.8: The adaptive area of ALURP

The data dissemination process of ALURP is much like LURP as well. Sensor data is first forwarded from the source node to the destination area. When the data packet reaches the dissemination node, it is further forwarded within the destination area to the mobile sink. Although some sensor nodes in the destination area of ALURP may not always be updated, they will still have valid routes to the sink, because the routing paths of these nodes will eventually lead the data to some nodes inside the adaptive area which have the newest information about the mobile sink. There is one exception, however. When the moving direction of the sink is approaching the VC and the radius of the adaptive area is shrinking, a problem arises that sensor nodes in the former adaptive area but not in the current one will still forward their data to the previous position of the sink. For example, as illustrated in Figure. 2.9, the previous position of the sink was at Pos B and it now moves to Pos C. As Node A is not inside the current adaptive area (Area C), it does not get the newest location updates from the sink. Therefore, Node A will still try to forward sensor data through Node B and Node C to the mobile sink, whereas neither of these two nodes belongs to the current adaptive area (Area C). As a result, a few subsequent data messages will be lost. In order to resolve this problem, the

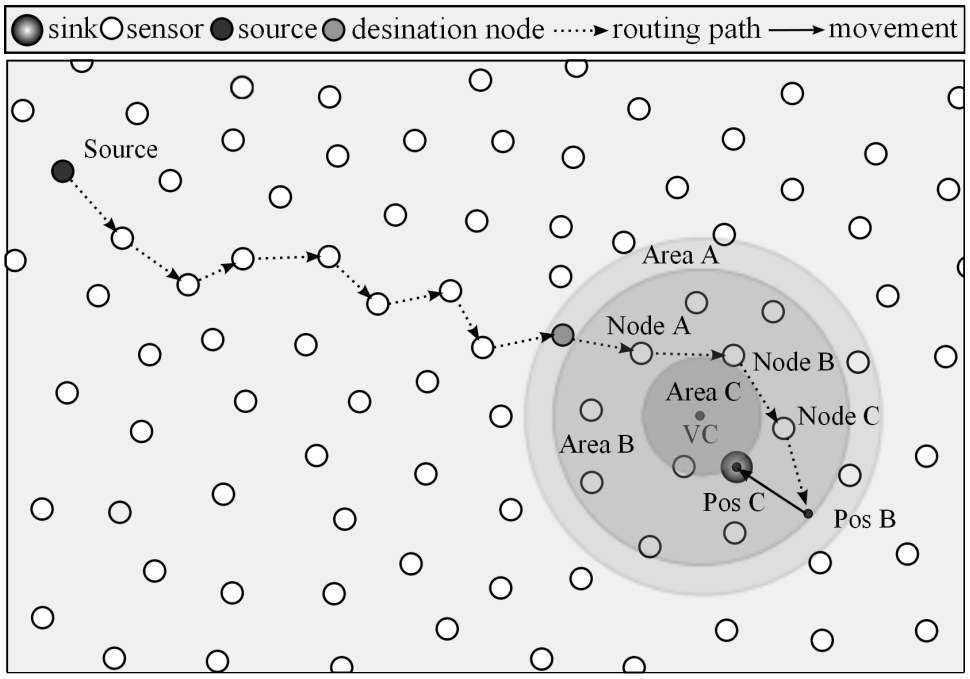


Figure 2.9: An exception scenario of the adaptive area

mobile sink should broadcast its location to sensor nodes which are located in the former adaptive area but not in the current adaptive area, *i.e.*, if the current adaptive area is smaller than the former adaptive area, the sink will send its location updates to the former adaptive area. The update algorithm of the mobile sink in ALURP is presented in Figure. 2.10.

The ALURP protocol further improves the energy efficiency of its predecessor, the LURP protocol, with a slightly longer data delay. To our knowledge, it is the first mobile sink protocol that comes up with an on-the-fly broadcasting range adjusting mechanism. Still, although the local update range of this protocol is to some extent adaptive to the moving speed of the mobile sink, ALURP has not clearly stated how to find a proper size or radius for the destination area either. Without clarifying this particular issue, the performances of both ALURP and LURP are unstable and depend heavily on different network topologies and application scenarios.

```
// initially when the mobile sink starts
broadcast global location updates;
construct a destination area centered at itself;
adaptiveRadius = 0;
start checkPostionTimer;
WHILE the mobile sink is moving within the sensor field
   IF checkPostionTimer expires THEN
      distanceToVC = distance(currentPos, VCPos);
      // if the sink moves out of the destination area
      IF distanceToVC > RADIUS THEN
         broadcast global location updates;
         construct a new destination area;
         adaptiveRadius = 0;
         reset checkPostionTimer;

      // if the sink is still in the destination area
      ELSE
         IF distanceToVC > adaptiveRadius THEN
            adaptiveRadius = distanceToVC;
         ENDIF
         broadcast local updates to the adaptive area of
                   radius adaptiveRadius centered at VC;
         reset checkPostionTimer;
      ENDIFELSE
   ENDIF
ENDWHILE
```

Figure 2.10: Pseudo code of the mobile sink in ALURP

2.2.5 The Two-Tier Data Dissemination Protocol

The Two-Tier Data Dissemination protocol (TTDD) proposed by Luo et al. [31] provides scalable and efficient data dissemination to multiple mobile sinks in the large sensor networks. Unlike the flat protocols described in the previous part of this chapter (Dual-Sink, Efficient Routing, LURP and ALURP), TTDD utilizes a virtual grid to eliminate the frequent sink location updates. Only the sensor nodes located at the crossing points of the grid, which are also called the dissemination nodes, participate in the data dissemination process of TTDD. Upon the detection of an event, the source node does not attempt to forward event data to the mobile sink immediately; instead, it proactively builds a grid structure throughout the sensor field. With this virtual grid in place, data queries sent from the mobile sink are able to traverse two tiers to reach the source. The lower tier is in a cell-size local grid square of the sink's current position. When the mobile sink requests event data, it first broadcasts a data query within the local cell. The higher tier is among the dissemination nodes along the grid. When a nearby dissemination node receives the data query, it forwards the query to its upstream dissemination node toward the source, which in turn further forwards the query upstream until the source is reached. The query forwarding process helps to build a path on the virtual grid between the mobile sink and the source so that event data can be forwarded through the same two tiers and follow the same routing path from the source back to the sink in a reverse direction.

The data dissemination process in TTDD involves three phases: the grid construction, the query forwarding and the data forwarding. In the grid construction, when the source node detects an event and generates corresponding event data, it builds a virtual grid throughout the sensor field with its own position as the start-crossing point (see Figure. 2.11). The source node then propagates a data announcement message which contains the information about the event and the source along the grid. Each data announcement message stops at a node closest to the next grid crossing point specified in the message (*e.g.*, Node A). The node stores the information about the source, and forwards the message to all its neighbor crossing points, except the one from which it receives the message. After this recursive propagation of data announcement messages, the grid construction phase is completed and the sensor nodes closest to the crossing points of the virtual grid become the dissemination nodes of that source.

The query forwarding phase starts when a mobile sink requests event data. The sink broadcasts a data query within its own local cell. The query will be received by

a local nearby dissemination node, which will further forward it along the virtual grid to the upstream dissemination node from which the local dissemination node receives data announcement messages. The upstream dissemination node continues to forward the query upstream towards the source until the query message finally arrives at the source. During this query forwarding process, each dissemination node that the query has passed through records the location information of its downstream dissemination node from which it receives the sink query, and uses that node as its next hop for data delivery in the following data forwarding phase.

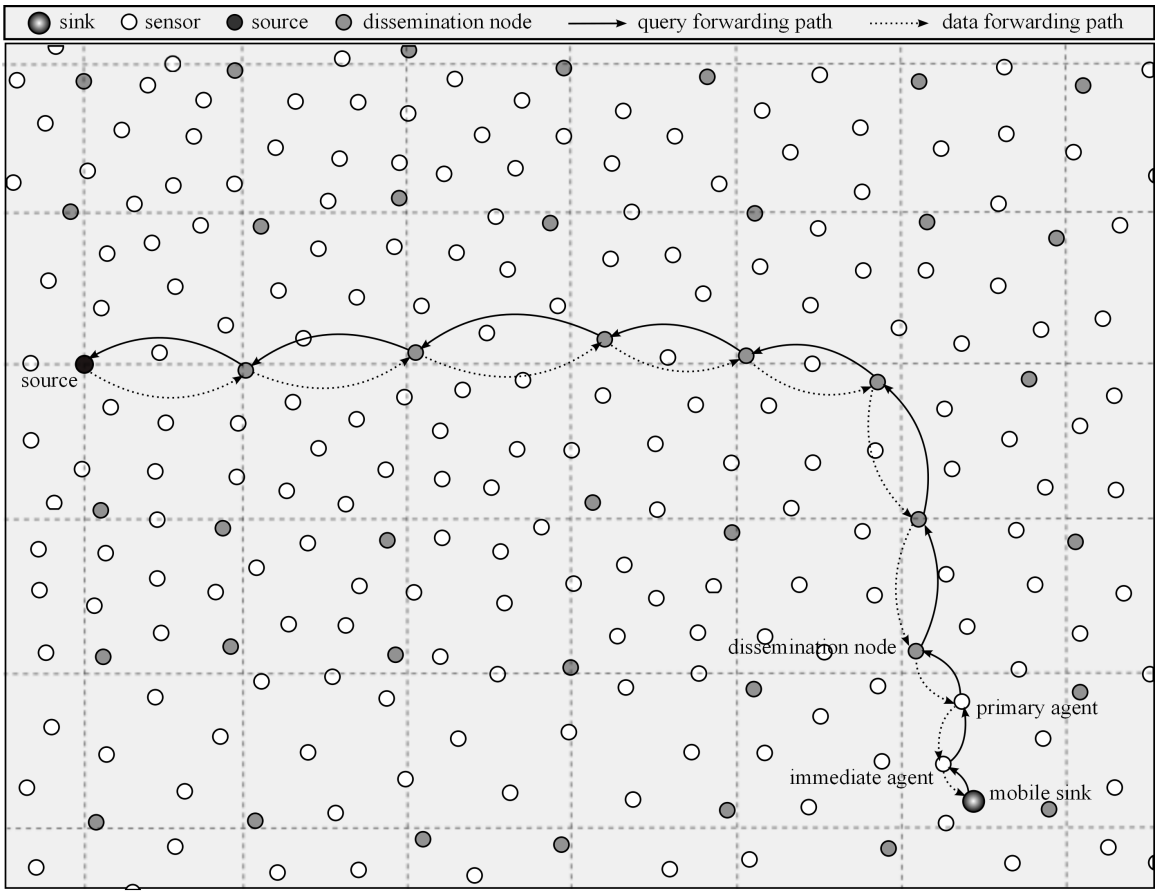


Figure 2.11: Overview of the data dissemination in TTDD

Once the source node receives the sink query from one of its neighboring dissemination nodes, it starts to send the requested event data to that dissemination node, which subsequently forwards the data downstream hop by hop along the virtual grid to the original local dissemination node which receives data queries from the mobile sink

directly. Then, the local dissemination node relays data to the mobile sink. The sink might be continuously moving when it receives data from the local dissemination node; however, sink mobility has been made transparent to the local dissemination node by a data forwarding technique called the trajectory forwarding. In the trajectory forwarding, the mobile sink first selects a sensor nearby that is not a dissemination node as its primary agent. The location of the primary agent is propagated in the sink queries. Thus, the primary agent is representing the mobile sink at the local dissemination node and the local dissemination node holds the responsibility to deliver data to the primary agent only. The rest of the work is left to the trajectory forwarding. The mobile sink employs an immediate agent as well, which is at first the same node as the primary agent, but later switches to some other node when the sink moves. Once the mobile sink is about to move out of the transmission range of its current immediate agent, it selects another neighbor sensor as its new immediate agent and sends the location of the new immediate agent to its primary agent. Therefore, no matter how far the mobile sink has moved away from its primary agent, the primary agent is always able to deliver the data from the local dissemination node to the current immediate agent of the sink, which then forwards the data to the sink directly. Figure. 2.11 illustrates an overview of the data dissemination process in TTDD: the query forwarding, the data forwarding along the virtual grid, and the trajectory forwarding within the local grid cell.

The TTDD protocol is a virtual infrastructure based data dissemination protocol targeting at large-scale sensor networks. It adopts an event-driven and query-based application scheme. Because the query broadcasting of the mobile sink is constrained within a local grid cell only, TTDD manages to avoid the excessive overhead and energy consumption of network-wide broadcasting over large-scale sensor networks through the use of a virtual grid per data source. Only the dissemination nodes on the crossing points of the virtual grid participate in the data dissemination, and other nodes are relived from transmitting data packets or location update messages. Therefore, TTDD can scale with extremely large sensor networks and the network size does not necessarily affect the performance of this protocol. The implementation of TTDD requires sensor nodes to be aware of their own locations through GPS signals or some virtual coordinate systems. It is a valid assumption, since in many applications the sensors are designed to know their geographical locations in order to tag the event data. TTDD proposes to build the virtual grid on a per source basis, in which different source nodes are associated with different sets of dissemination nodes. The authors of this protocol claim such design can scale to a large number of sources and provide load balancing. However, Jeon et

al. in [24] argue that if the number of source nodes is increased, the communication overhead and storage cost of the network system can be considerably higher due to the maintenance of the virtual grid structure and the dissemination nodes for each source. Jeon et al. further prove in their simulation results that TTDD consumes more sensor energy than any other selected protocol and does not have a very competitive network lifetime performance.

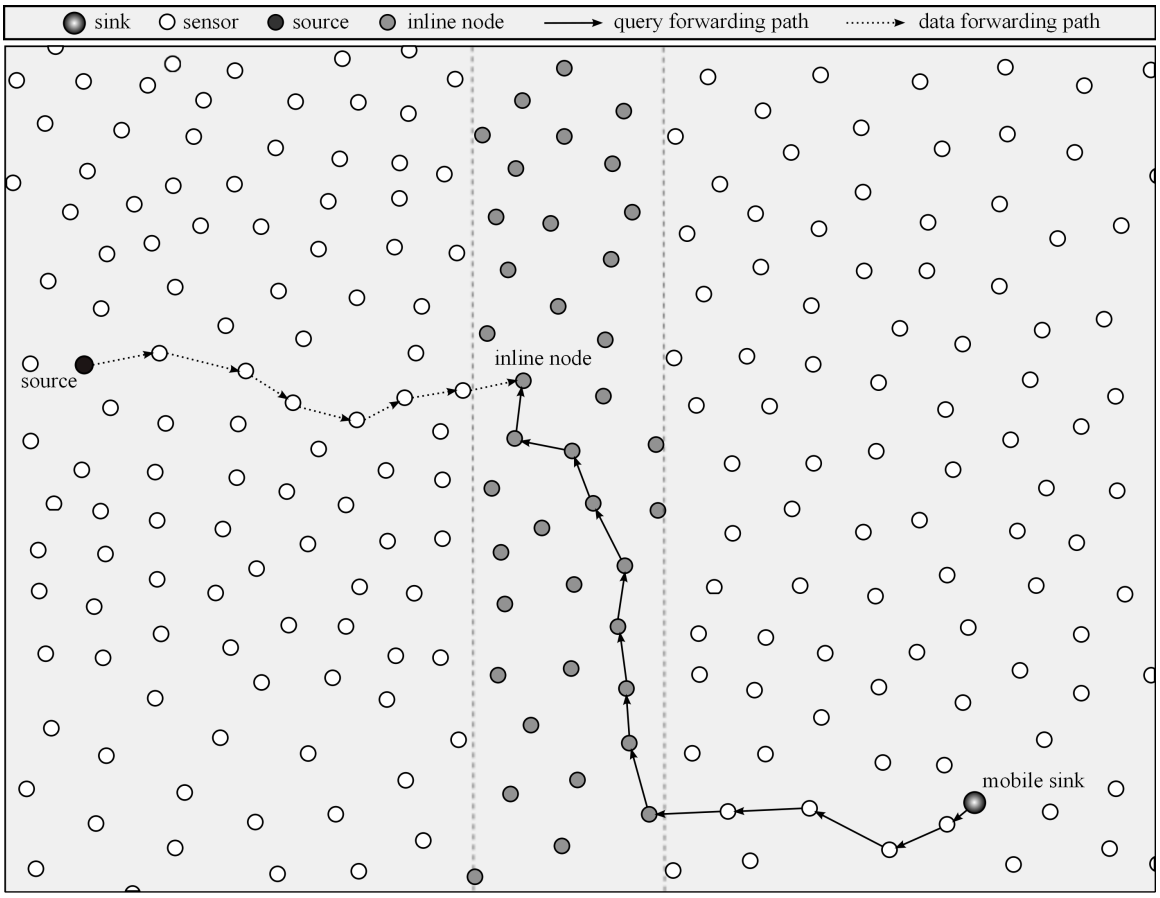


Figure 2.12: Overview of the data dissemination in LBDD

2.2.6 The Line-Based Data Dissemination Protocol

The Line-Based Data Dissemination protocol (LBDD) [18] is also a virtual infrastructure based protocol for WSNs with mobile sinks. LBDD uses a virtual vertical line or strip area (see Figure. 2.12) constructed in the central field of the wireless sensor network

that divides the network into two equal parts. Sensor nodes inside the boundaries of this virtual area are called the inline nodes. The line or strip area acts as a rendezvous area for sensor data and sink queries. Upon the detection of an event, the source transmits the gathered data toward the virtual area in the center of the network. The data is stored on the first inline node encountered on the forwarding path from the source to the rendezvous area. When the mobile sink requests event data, it also sends a query toward the rendezvous area. Once inside the rendezvous area, the sink query travels linearly until it encounters the data storing inline node. Then, based on the location information of the mobile sink propagated in the sink query, event data is geographically routed to the sink.

2.2.7 The Railroad Protocol

The Railroad protocol proposed by Shin et al. [42] adopts a concept similar to the LBDD protocol. It virtually constructs a square-shape strip area that is called the rail in the central area of the sensor network. Sensor nodes located inside this strip area become the rail nodes. When the source is triggered by a certain event, it stores the event data locally and sends the corresponding meta-data (*e.g.*, a data announcement message) toward the nearest rail node. The mobile sink sends its data queries to the rail as well. Once inside the rail, the query subsequently travels around the square-shape strip area until it arrives at a rail node that holds the meta-data about the event and the related source node. Event data is then directed from the source to the mobile sink geographically based on the source location information provided in the meta-data and the sink location information provided in the sink query.

The common drawback of the Railroad protocol and the LBDD protocol (and other similar virtual infrastructure based protocols) is that they both use a fixed virtual area in the sensor network to assist the data dissemination process. In homogeneous WSNs where all deployed sensors have the same energy storages, protocols like Railroad and LBDD are usually not applicable due to the hot spot problem caused by the fixed rendezvous area in the sensor field.

2.2.8 The Coordination-based Data Dissemination Protocol

The COordination-based data Dissemination protocol for wireless sEnsor networks (CODE) [50] is a novel virtual grid based protocol which employs a sleep-wake mechanism. Researchers have found that the sensor energy consumed during the idling state cannot

be ignored [40]. Therefore, allowing the sensor nodes that do not necessarily have to participate in the data dissemination process to sleep for a short period of time might be a smart choice to save sensor energy and improve the network lifetime. In the CODE protocol, the sensor area is partitioned into virtual grids that are indexed based on their geographical locations. The grids are partitioned in a way that all nodes belonging to the same grid function equally, *i.e.*, any node from one grid is able to communicate directly with all the nodes in the neighboring grids. One coordinator can thus be selected to represent an entire grid and act as an intermediate node to cache and relay data in the sensor network. The other sensor nodes within the grid are then considered redundant and allowed to sleep for a while. Each coordinator node is replaced by another node from the same grid if its remaining energy is below a threshold.

To establish a data forwarding path from the source to the mobile sink, CODE uses a similar query and data forwarding scheme to the TTDD protocol discussed in the previous part of this section. Data queries are forwarded upstream from the mobile sink to the source through a series of selected coordinators, and event data is forwarded downstream, following the same path but in the reverse direction, back to the sink. A slight difference in CODE is that the routing algorithm is based on grid IDs rather than individual node locations, *i.e.*, without knowing which sensor node is the coordinator of the next virtual grid on the routing path, the intermediate coordinator node only needs to figure out which grid it should relay data to next.

The CODE protocol is a virtual infrastructure based data dissemination protocol for WSNs with mobile sinks. Like most of the virtual infrastructure based protocols, the sensor nodes in CODE are event-driven, and the mobile sink must query the source for specific event data. Also, the sensors deployed in CODE are required to be location-aware. The major contribution of this protocol is the sleep-wake mechanism it proposes, targeting the unnecessary sensor energy consumption during the idling state. As a matter of fact, we get inspirations from this novel design to develop our own data dissemination protocol stated in this thesis. However, as sensor nodes are usually distributed irregularly in the sensor field in real world applications, the virtual grid partitioning, which is one of the key processes of the sleep-wake mechanism in CODE, is faced with considerable difficulties.

2.3 Summary

Thus far, we have engaged with relevant works related to the central concerns of this thesis. A number of major data dissemination protocols for WSNs with mobile sinks have been investigated in this chapter. A comparison of these protocols based on the discussion of protocol classifications in Section 2.1 is presented in Table. 2.2. As we can tell, considerable effort has been made to design and implement fast, reliable and energy efficient protocols to accommodate the sink mobility in WSNs. Since the hot spot problem in static sink approaches is addressed by introducing mobile sinks to the sensor network, mobile sink protocols mainly focus on how to efficiently track the movement of the sink, or, more specifically, how to reduce the overhead and energy consumption of sink location updates.

Protocol	**Application Type**	**Disseminated Information**	**Sink Mobility**	**Virtual Structure**	**Location Awareness**
Dual-Sink	Continuous	Sink Location & Data	Predefined	None	No
Efficient Routing	Continuous	Sink Location & Data	Random	None	No
LURP	Continuous	Sink Location & Data	Random	None	Yes
ALURP	Continuous	Sink Location & Data	Random	None	Yes
TTDD	Event / Query	Sink Location, Meta-data & Data	Random	Grid	Yes
LBDD	Event / Query	Sink Location & Data	Random	Strip	Yes
Railroad	Event / Query	Sink Location, Meta-data & Data	Random	Railroad	Yes
CODE	Event / Query	Sink Location & Data	Random	Grid	Yes

Table 2.2: Comparison of the data dissemination protocols

Well designed flat data dissemination protocols usually constrain the sink location updates within a very limited sensor area rather than through the entire network. Dual-Sink manually sets the beacons of the mobile sink a small TTL; Efficient Routing uses a threshold to measure the change in cost and lets the sensors decide whether or not to forward the update messages; LURP and ALURP incorporate a concept of the destination area to constrain the location broadcasting in this local area only. Virtual infrastructure based protocols organize the underlying physical network into virtual structures such as grids, strips, circles or clusters, so that the sink queries, location updates and sensor data can be conveyed on these overlay virtual structures without the participation of other nodes that do not belong to the virtual infrastructure in the network. The overhead of the sink location updates is greatly reduced in this way. Meanwhile, since only the sensors located on the virtual structures participate in the sink query forwarding and data forwarding processes, the virtual infrastructure based protocols can scale with very large sensor networks without a significant performance degradation in energy efficiency.

However, there is one issue that most of the current protocols have ignored. The idling energy consumption of sensor nodes actually dominates the overall energy consumption and the network lifetime of WSNs. Without paying attention to the idling energy consumption, none of the protocols can achieve optimal energy efficiency. The CODE protocol introduces a sleep-wake mechanism to reduce the idling energy consumption, but the method to equivalently partition the network into virtual grids in CODE is not practical in real world applications. Therefore, we came up with the E-Trail protocol, a trail-based flat data dissemination protocol which also employs a novel sleep-wake mechanism different from CODE. E-Trail will be discussed in detail in Chapter 3.

Chapter 3

E-TRAIL: An Energy Efficient Trail-Based Protocol

A number of popular data dissemination protocols for WSNs with mobile sinks such as Dual-Sink [48], Efficient Routing [15], LURP [46], ALURP [47] and TTDD [31] have been reviewed in Chapter 2. The protocols each develop their own mechanisms to maintain the network connectivity with the mobile sink and reduce the overhead and energy consumption of the sink location updates. However, despite their contributions, most of the above protocols face different kinds of problems. Therefore, we are motivated to design and implement E-Trail, a novel Energy efficient TRAIL-based data dissemination protocol, which carries forward the strengths of the major existing mobile sink protocols and also avoids the shortcomings already found in these protocols. The contribution of E-Trail is the combination of a trail-based mechanism and a sleep-wake scheme to further improve the network lifetime. In this chapter, a detailed discussion of the proposed E-Trail protocol will be presented in terms of the motivation behind this protocol, its important technical features, and its design and implementation.

3.1 Motivation

Mobile sink protocols of WSNs can be classified as either flat data dissemination protocols or virtual infrastructure based data dissemination protocols, based on their network routing architectures. Among the eight protocols investigated in Chapter 2, four (Dual-Sink, Efficient Routing, LURP and ALURP) are flat protocols and the other four (TTDD, LBDD, Railroad, CODE) are virtual infrastructure based protocols. Hamida

et al. claim that overlaying a virtual infrastructure over the physical network topology has often been regarded as a more efficient strategy than the flat architectures and the flooding based protocols for effective data dissemination in WSNs with mobile sinks [19]. However, according to our experimental results, this is not always the case. Recently, as more and more flat approaches are able to constrain the location broadcasting of the mobile sink among a very small neighborhood of sensors during the entire network lifetime (or at least for most of the time during the network operation), the virtual infrastructure based protocols no longer outperform the flat protocols in energy efficiency, data delay and other network metrics. In fact, the energy performance of well-designed flat mobile sink protocols such as Efficient Routing and ALURP are much better than typical virtual infrastructure based protocols such as TTDD, especially in small or medium sensor networks (*e.g.*, less than 1000 nodes are deployed in the sensor field). This is usually because the high maintenance cost of the virtual infrastructure across the entire network and the complexity of the query-based data gathering scheme can offset and sometimes even exceed the energy gain from eliminating the network-wide broadcasting in the network. Nevertheless, the virtual infrastructure based protocols still have their advantages in large-scale sensor networks, as their energy performance hardly degrades with the network size. In contrast, the flat data dissemination protocols do not scale with very large networks. The network-wide broadcasting of the flat protocols can be overwhelming and cause a lot of network collisions and packets lost in the large-scale sensor networks, although the frequency of the network-wide broadcasting in many protocols are quite low today.

E-Trail is targeted at medium sensor networks. Therefore, a flat architecture based on flooding is preferred. Recalling the four flat mobile sink protocols discussed in Chapter 2, the Dual-Sink protocol is a rather immature design. It restricts the movement of the mobile sink and limits the number of hops that a sink beacon can spread to, so that the network lifetime gains from the sink mobility will not be offset by the high energy loss in frequent network-wide broadcasting. Dual-Sink uses a static sink as well. When the nodes do not know where the mobile sink is, they send their data to the static sink directly. However, as the beacon range of the mobile sink is very small, when the network size becomes large enough so that the data collection of the mobile sink does not have a notable impact on the overall data dissemination in the network, *i.e.*, when most data packets are directed to the static sink, the Dual-Sink degrades to a protocol more like a pure static sink approach.

Efficient Routing, LURP and ALURP all share a common problem: their energy

performance depends heavily on certain important parameters which are not clearly stated in the literature regarding how to find the proper values of these parameters. For instance, the overhead of location updates in Efficient Routing is mainly determined by a threshold that measures the change between the registered cost at sensor nodes and the actual cost propagated in the update messages. Similarly, LURP and ALURP both do not have full control over when the mobile sink should initiate a new round of network-wide broadcasting; it is rather determined on the fly based on whether the mobile sink has moved out of the current destination area. Thus, the radius of the destination area is critical to the performance of the two protocols. Moreover, none of the protocols have put effort into reducing the unnecessary idling energy consumption of sensor nodes to further extend the network lifetime. Thereby, we propose a new protocol for data dissemination in WSNs with mobile sinks: the Energy efficient TRAIL-based data dissemination protocol (E-Trail), which not only provides reliable and fast data dissemination to the mobile sink without much broadcasting, but also incorporates a sleep-wake mechanism to further reduce the energy consumption by putting unnecessary sensor nodes into sleep mode for a period of time. With the combination of the trail-based approach and the sleep-wake mechanism, E-Trail saves nearly half of the average energy consumption, compared to the other selected protocols, and significantly improves the network lifetime. More details of the E-Trail protocol will be introduced in the later sections of this chapter. Then, the performance evaluation of E-Trail will be presented in Chapter 4.

3.2 Important Features

As a human body walks through the woods or wilderness, a marked or beaten path will be left by his or her feet. One important feature of the proposed E-Trail protocol relies on this concept. The mobile sink sends the location update messages (also called the trail beacons in E-Trail) periodically as it moves through the sensor field. The nearby sensor nodes receiving the trail beacons build a trail which leads to the current position of the sink. The idea of leaving a trail of bread crumbs is not new, and it was first exploited by routing protocols for mobile ad hoc networks [36]. In E-Trail, the mobile sink becomes the cluster head when it builds a cluster of sensor nodes at the beginning of every round. As the sink moves away from the initial position where it built the cluster, a trail is left by the sink through the broadcasting of the trail beacons. The trail-based approach enables each node from the cluster to find the current location of the sink and

forward data to it. The trail generation algorithm is completely local and it introduces minimal overhead on the tracking of the mobile sink and the maintenance of the routing paths. In addition, the sleep-wake mechanism takes place after the mobile sink builds a cluster. The basic concept of the sleep-wake mechanism is to let every single node in the cluster sleep, unless some sensor node tells it not to. The goal is to conserve the sensor energy in sleep mode, while each sensor node in the network (no matter if it sleeps or not) still has a valid routing path consisting of awake nodes to deliver data to the mobile sink. In case of the failure of a routing path, the path recovery can be triggered. To this end, the E-Trail protocol comprises at least four important technical features: the cluster formation, the sleep-wake mechanism, the trail generation, and the path recovery. All the features will be explained in detail in this section.

3.2.1 The Cluster Formation

The E-Trail protocol inherits a cluster formation mechanism from its precursor, the MDC/PEQ protocol [38]. The mobile sink uses restricted broadcasting with a limited TTL (Time-To-Live) to build a cluster. Clusters are created in rounds. A round has been defined as a specific period of time (*e.g.*, 50 seconds) during the network operation to simulate a reset of all the routing paths in the network and prevent the trail left by the mobile sink from detouring too far.

The mobile sink builds a cluster by broadcasting a cluster configuration message (CLU_CFG) with a specific TTL to the sensor network. The sink itself then becomes the cluster head. The CLU_CFG message carries the ID of the mobile sink, the intermediate node that forwards the message, the remaining energy of the intermediate node, the hop count, and the age of the cluster. The ID of the mobile sink identifies which sink generates the CLU_CFG message. As we assume multiple mobile sinks are moving within the sensor field, multiple clusters can exist simultaneously in the network. The ID of the mobile sink enables the sensor node to record its cluster head so that it knows which cluster it belongs to. The intermediate node is the node which directly forwards the CLU_CFG message from the upper stream to the current node. It can be one of the next hop candidates for the current node to deliver data towards the mobile sink. The remaining energy of the intermediate node reflects the residual energy left in the battery of the intermediate node. The hop count is the number of hops that the CLU_CFG message has passed through, which indicates the hop distance from the current node to the mobile sink. The age of the cluster is equivalent to the period of a round. When a

cluster expires, the sensor nodes of the cluster remove all their routing entries from the routing tables. As a result, the routing paths of the cluster are all cleared.

By checking the hop count, a sensor node that receives a CLU_CFG message decides whether the intermediate node of the message could be its next hop candidate. If the hop count is greater than the hop distance previously stored in the routing table, the node will simply discard the CLU_CFG message. If the hop count is equal to the current hop distance, the node will add a new routing entry composed of the address of the intermediate node, the hop distance and the remaining energy to its routing table. Then, the node does not forward the CLU_CFG message any further, because this helps to reduce the unnecessary overhead of broadcasting. If the hop count is smaller than the hop distance recorded, the sensor node will clear its routing table first and add the intermediate node as a new entry. Then, the node increments the hop count of the CLU_CFG message by one and forwards the message. The message is continuously forwarded this way until its hop count reaches the specified TTL. Sometimes it happens that a node that has already joined a cluster receives a CLU_CFG message from another sink. Under this circumstance, the node will either decide to join the new cluster, or to stay with the current cluster, based on the hop distance to its current cluster head and the hop count propagated in the CLU_CFG message. If the node decides to stay in the same cluster, the CLU_CFG from another sink is simply dropped and will not be forwarded any further.

At the end of the cluster formation phase, each node in the cluster has a routing table consisting of all its next hop candidates to deliver data and their corresponding residual energy. This routing information is valid until another round of the cluster creation is initiated by the sink. In our work, we use a relatively large TTL for the CLU_CFG messages to make sure, after one round of the cluster formation by multiple mobile sinks in the sensor field, that each node in the network belongs to a cluster and none of them is left behind.

3.2.2 The Sleep-Wake Mechanism

After the cluster formation phase each clustered node will have a list of neighboring nodes through which it can forward data to the mobile sink. One node can have multiple neighbors in its routing table, depending on the node density. The neighboring nodes are called the next hop candidates, as they all have the same hop distance to the cluster head and only vary in their residual energy. However, based on our observations, a sensor

node only needs one of its neighbors per round to deliver data to the cluster head. We can thus put some of the neighboring nodes into sleep mode during the rest period of the current round without breaking any routing paths, as long as the sleeping nodes can wake up in time just before the next round of the cluster formation process begins.

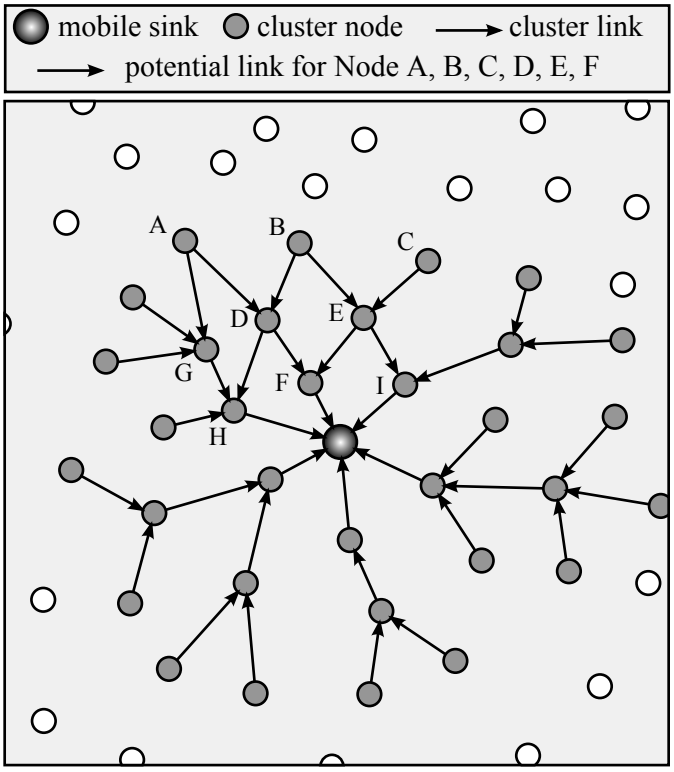


Figure 3.1: The propagation of the NO_SLEEP messages

E-Trail uses the remaining energy of the next hop candidates in the routing table to decide which ones could go to sleep and which ones should be awake. During the cluster formation phase, each node starts a timer immediately when it receives a CLU_CFG message. If a clustered sensor node does not receive any NO_SLEEP message from its neighboring nodes before the timer expires, the node will turn its radio off and switch to sleep mode for a period of time that is usually slightly shorter than a round. The NO_SLEEP message is a one hop message which indicates the sender of the message will need the destination node of the message to deliver data toward the sink. According to the design of E-Trail, a sensor node will send a NO_SLEEP message to its next hop candidate with the most residual energy in the routing table. In Figure. 3.1, if we set the TTL of the CLU_CFG messages as small as 3 just to illustrate the sleep-wake mechanism,

IntermediateNode	**HopDistance**	**ResidualEnergy**
G	3	86.93
D	3	92.75

(a) Routing table of Node A

IntermediateNode	**HopDistance**	**ResidualEnergy**
D	3	92.75
E	3	90.36

(b) Routing table of Node B

IntermediateNode	**HopDistance**	**ResidualEnergy**
E	3	90.36

(c) Routing table of Node C

IntermediateNode	**HopDistance**	**ResidualEnergy**
H	2	79.83
F	2	82.16

(d) Routing table of Node D

IntermediateNode	**HopDistance**	**ResidualEnergy**
F	2	82.16
I	2	80.37

(e) Routing table of Node E

Table 3.1: Routing tables of the corresponding sensor nodes

then after the cluster creation process of the mobile sink, 33 nodes are in the cluster. The routing tables of Node A, Node B, Node C, Node D and Node E are listed in Table. 3.1, respectively, except that we fabricate the values of the residual energy of each sensor. As a result, Node A will send a NO_SLEEP message to Node D. Node B will also send a NO_SLEEP message to Node D because Node D apparently has more energy remaining. Similarly, Node C will send NO_SLEEP to its only next hop candidate Node E, Node D will send NO_SLEEP to Node F, and Node E will send NO_SLEEP to Node F as well. When the propagation of the NO_SLEEP messages is finished, those sensor nodes (*e.g.*, Node A, Node B and Node C) that have not received any NO_SLEEP message before the expiration of the timer set after the reception of the CLU_CFG messages can turn their radio off and sleep for a while. If we use hexagons to mark the nodes which are allowed to sleep in the figure, the network topology with sleeping nodes will more or less look like the one presented in Figure. 3.2. Note that although a very small cluster is used here to demonstrate the sleep-wake mechanism, more than half of the clustered nodes (19 of 33) can switch to sleep mode for the rest of the current round. Such performance is indeed impressive.

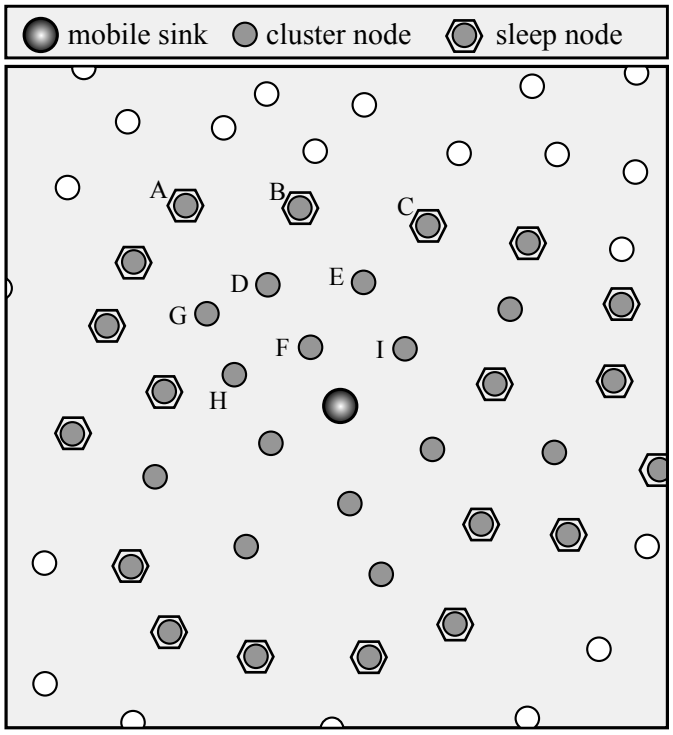


Figure 3.2: Network topology of E-Trail with sleeping nodes

The sleep-wake mechanism not only saves the sensor energy, but also balances the energy consumption after a few rounds of operation. The sensor nodes that sleep in one round will eventually be awake in some future rounds because they have consumed less energy in the sleep mode than other nodes. The sleeping period should not exceed the round duration, and the sleeping nodes should wake up before the next cluster formation begins. The sleep-wake mechanism of E-Trail guarantees that each sensor node in the cluster has at least one neighboring node awake to forward data toward the sink. A question might arise here about what happens when a sensor node that is supposed to be triggered by certain events goes to sleep. Regarding this issue, we assume that any sensor node will be able to wake up its radio whenever an event is detected by its sensor module. This assumption is valid, since a number of hardware solutions (*e.g.*, the iMote2 IPR2400 processing and sensing module [3]) currently implement such wake-up mechanisms.

3.2.3 The Trail Generation

The clustered sensor node learns a path to deliver data to the cluster head from the information provided in the CLU_CFG messages. However, as the mobile sink will eventually move out of the range of its immediate neighbor, the link between the cluster and the cluster head will be disconnected. Reducing the interval of sending the CLU_CFG messages to only a few seconds would definitely solve the problem; however, it makes the E-Trail protocol no better than the other flooding based protocols. Thereby, to minimize the overhead, we use a new type of beacon — the trail beacons to update the locations of the mobile sink during a round.

As the mobile sink moves away from the initial position where it built the cluster, it periodically sends out the TRAIL_BEACON messages with a very small TTL (TTL = 2 has shown a good balance between the overhead and the energy efficiency). Using TTL = 2 for instance, only sensor nodes that are less than two hops away from the mobile sink will be updated by the trail beacons. The result after several rounds of the trail beacons is a trail left by the movement of the sink (see Figure. 3.3). The time interval for the broadcasting of the TRAIL_BEACON messages should be set to a proper value to accommodate the movement of the sink, *i.e.*, it should be adaptable to the sink's moving speed. If the time interval is set too large, the sink could have moved far away from its immediate ex-neighbor node and the sending of a new TRAIL_BEACON message would not be enough to maintain the trail connected. On the other hand, if it is set too

small, unnecessary update overhead would be produced and more sensor energy would be wasted.

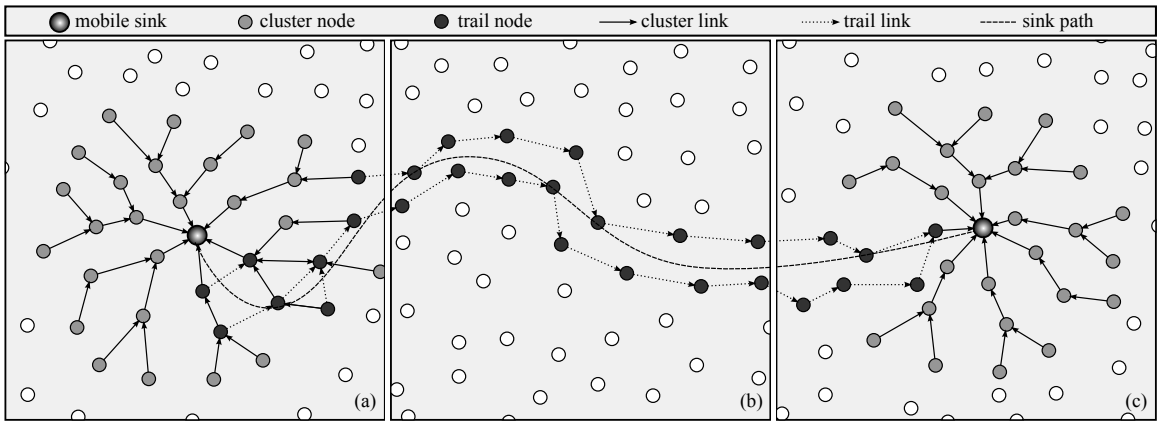


Figure 3.3: Overview of the mobile sink activities in E-Trail: (a) the mobile sink starts a cluster with $ttl = 3$. Nodes learn the paths to the sink. (b) eventually, the mobile sink moves and sends trail beacons periodically to build a trail. The trail ensures that sensor nodes will still have a valid path to the sink after it moves. (c) a round finishes and the mobile sink restarts to create a new cluster.

The TRAIL_BEACON message carries similar information to the CLU_CFG message: the ID of the mobile sink, the intermediate node that forwards the message, and the hop count. However, the routing updates of sensor nodes upon the reception of the trail beacons are not necessarily based on the hop counts. The hop count propagated in the TRAIL_BEACON message is only used to limit the spread of this message. When the sensor node receives a TRAIL_BEACON message, it first determines whether it has already received the trail beacons of a same round. If so, the TRAIL_BEACON message is simply discarded to prevent recursive message forwarding. Otherwise, the node clears its routing table, sets its cluster head as the ID of the mobile sink propagated in the message, and adds a new routing entry. Then, the node increments the hop count by one and forwards the message to other nodes if the hop count does not reach the TTL of the trail beacons. Consequently, the routing table of a sensor node on the generated trail usually has only one entry, because it is updated only once during the same round of trail beacons and will be cleared before adding a new entry in the next round. Also, the recorded hop distance of the sensor nodes on the trail may not reflect the real hop distance to the mobile sink, as they are usually obsolete. For example, if the TTL of

the trail beacons is 2, the hop distance of the trail nodes can only be 1 or 2. The trails generated in this way are only valid for the duration of a round. The trail nodes will remove their routing tables when the age of the trail expires. Figure. 3.3 presents the overview of a round of the mobile sink activities in E-Trail, which begins with a cluster formation phase, proceeds with the trail generation, and ends with a new cluster formation phase.

3.2.4 The Path Recovery

Path breaks will eventually happen in E-Trail due to a variety of reasons:

- A sensor node on the routing path runs out of energy;
- A sensor node is malfunctioning;
- A sensor node shifts to the sleep mode whereas it is supposed to be awake, *e.g.*, the NO_SLEEP message from a neighboring node does not reach the destination node due to network collisions;
- The mobile sink is moving too fast and the frequency of the trail beacons is not enough to accommodate the sink mobility.

To this end, E-Trail implements an overhearing based path recovery mechanism to avoid consecutive packet lost. We assume the sensor node is able to overhear its neighboring nodes transmitting data packets within its transmission range, although the packets may not be directed to itself. Thus, if a node transmits data packets to its next hop and does not overhear the same data packet forwarded by its next hop for three consecutive data packets, it triggers the path recovery mechanism and broadcasts a HELP message to its one-hop neighbors. The HELP message carries the hop distance of the path breaking node to the mobile sink. A neighboring node will reply with a HELP_RESPONSE message if it has a route to the mobile sink. To avoid loops, the hop distance of the neighboring node should be less or equal to the hop distance propagated in the HELP message sent from the path breaking node. Furthermore, the neighbor should avoid sending HELP_RESPONSE messages to the node which is supposed to be its next hop on the routing path. Upon the reception of a HELP_RESPONSE message, the path breaking node adds the neighboring node as its new route. In case multiple HELP_RESPONSE messages are received from multiple neighbors, the path breaking node has two options: first, it could adopt the routing information provided in the first HELP_RESPONSE

message that arrives at the node; second, it could compare the hop distance of all the received HELP_RESPONSE messages and use the closest one to the mobile sink as its next hop.

There is one issue with the path recovery mechanism, however. Since the mobile sink does not forward data packets any further, the last sensor node on the routing path (the immediate ex-neighbor of the sink) will never be able to overhear the sink forward any packet. According to our investigations, path breaks are more likely to happen on these kind of nodes. Therefore, we have to let the mobile sink send back acknowledgements as a remedy. For the immediate ex-neighbor of the sink, once it transmits a data packet to the mobile sink, a timer is initiated. On receiving the data packet, the mobile sink sends back a SINK_ACK message. If the immediate neighbor node does not receive a SINK_ACK message before the timer expires, the routing path is considered broken. Then, the node triggers the path recovery mechanism and waits for the HELP_RESPONSE messages. The propagation of the SINK_ACK messages does not impose an extra burden on the sensor network, because the energy consumption of the mobile sink is usually not counted.

3.3 The Design and Implementation of E-Trail

The motivations and the important features of E-Trail have been explained in the previous sections of this chapter. From now on, we focus on the design and implementation of this protocol. Two types of wireless devices are involved in the data dissemination of E-Trail: the mobile sink and the sensor node. Thus, we discuss their design and implementation separately.

3.3.1 The Mobile Sink

The work flow of the mobile sink in E-Trail is graphed in Figure. 3.4. Generally, it consists of the following steps:

1. Initially, the mobile sink broadcasts a CLU_CFG message to the sensor network to build a cluster and becomes the cluster head. A round starts.

2. Every time when the trail beacon timer expires, the mobile sink sends out the trail beacons to the neighboring nodes and updates the generated trail.

3. When the duration of the round ends, the mobile sink initiates a new cluster formation process.

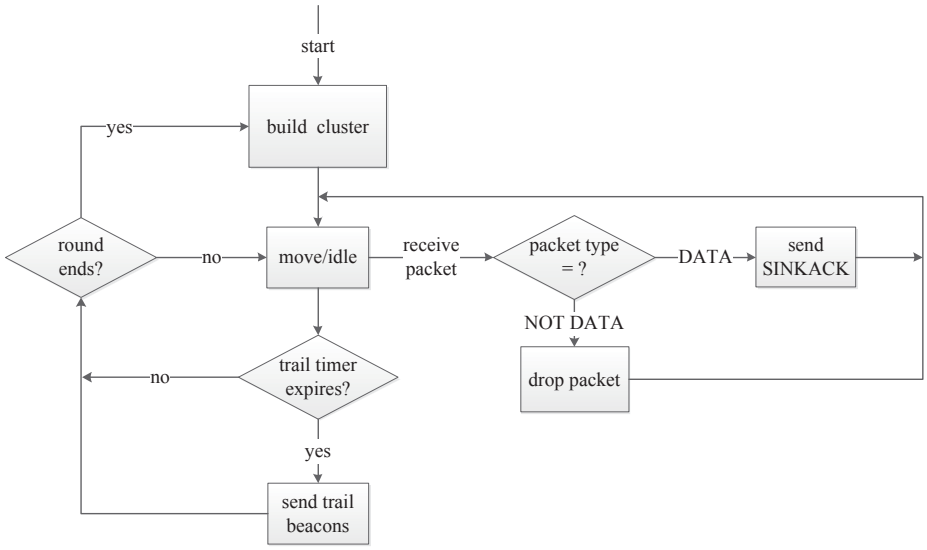


Figure 3.4: Work flow of the mobile sink in E-Trail

4. When the mobile sink is moving through the sensor field, it can receive new network packets. Usually there are only two possibilities: (1) it receives a DATA packet. The sink stores the sensor data in the DATA packet and sends back an acknowledgement. (2) it receives a packet which is not data. The packet is simply dropped.

Figure. 3.5 shows the corresponding algorithms of the mobile sink activities.

3.3.2 The Sensor Nodes

For most of the time, the sensor nodes in E-Trail only passively receive various types of messages and decide whether to forward them. However, in some situations they also actively send messages to other nodes. For example, the sensor node may send a NO_SLEEP message to its next hop candidate with the most residual energy, or it may broadcast a HELP message to its one-hop neighbors when the path recovery mechanism is triggered. When the node detects an event and becomes a source in the network, it starts to send data messages toward the mobile sink. The work flow of the sensor node in E-Trail is much more complex than the work flow of the mobile sink. Therefore, we have to use two graphs that complement each other in this thesis to represent the work flow of sensor nodes. Figure. 3.6 describes when a CLU_CFG message, a trail beacon or

```
PRELIMINARY roundTimer has been started for the mobile sink
WHILE the mobile sink is moving within the sensor field
    IF roundTimer expires THEN
        broadcast a CLU_CFG message;
        reset roundTimer;
        start trailTimer;
    ENDIF

    IF trailTimer expires THEN
        send TRAIL_BEACON messages;
        reset trailTimer;
    ENDIF

    IF a packet p is received THEN
        SWITCH(p.packetType)
            CASE DATA:
                store the sensor data provided in p;
                send a SINKACK message back;
            // if the received packet is not a data packet
            DEFAULT:
                drop the received packet;
        END SWITCH
    ENDIF
END WHILE
```

Figure 3.5: Pseudo code of the mobile sink in E-Trail

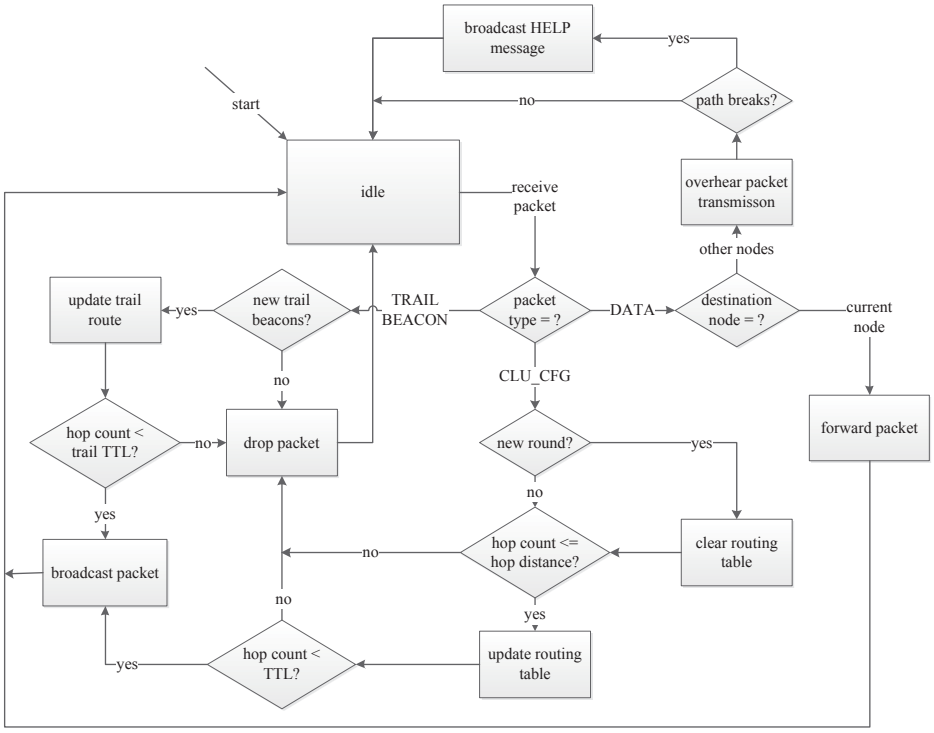


Figure 3.6: Work flow of the sensor node in E-Trail: Part A

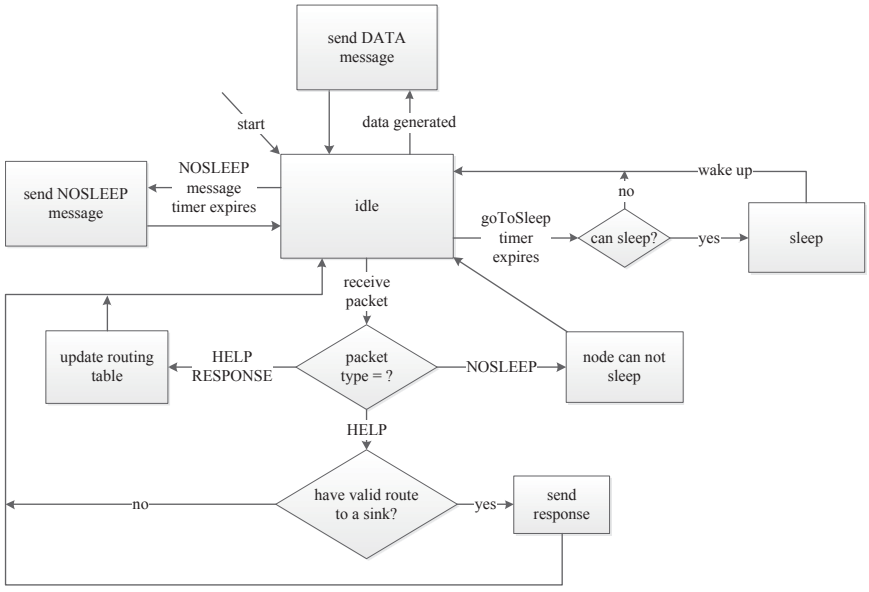


Figure 3.7: Work flow of the sensor node in E-Trail: Part B

a DATA message is received by the sensor, and Figure. 3.7 describes the rest. In brief, the two graphs cover the following procedures:

1. Initially, the sensor node is at idling state.

2. On receiving a CLU_CFG message, the node first decides whether the message is from a new cluster formation process or not. If a new round begins, the node clears its routing table and adds a new entry. If not, the node updates its routing table based on the hop count propagated in the message. Then, it adds the hop count by one and further forwards the message if the hop count does not reach the TTL of the CLU_CFG messages.

3. On receiving a TRAIL_BEACON message, the sensor node will only update its routing table if the trail beacon is new. Otherwise, the message is dropped. If the message is from a new round of trail beacons, the node increments the hop count by one and forwards the message until the hop count reaches the trail TTL.

4. On receiving a DATA message, the node first checks if the destination of the message is directed to itself. If so, the node looks up the next hop in its routing table and forwards the DATA message. Otherwise, it further checks if the source node of the message is the node to which it has forwarded a data packet. If that is the case, the node then has successfully overheard its next hop deliver the data packet to other nodes. On the other hand, if the node does not overhear its next hop forward the data packet for three consecutive times, it will consider the routing path has broken and trigger the path recovery mechanism.

5. On receiving a NO_SLEEP message, the sensor node is aware of that some other node is relying on it to deliver data to the mobile sink. So it will stay awake for the rest of the current round. Otherwise, if the node has not received any NO_SLEEP message when the goToSleep timer expires, it goes to sleep and will wake up before the next round starts.

6. On receiving a HELP message, the node sends back a response if it has newly updated routes to the mobile sink.

7. On receiving a HELP_RESPONSE message, if the node has requested help from the neighboring nodes, it updates its routing table based on the information of the message.

```
WHILE residualEnergy > 0
IF receive a packet p THEN
    SWITCH(p.packetType)
        CASE CLU_CFG:
            forwardFlag = false;
            // if it is a new round CLU_CFG message
            IF p.seqNo != cluSeqNo THEN
                cluSeqNo = p.seqNo;
                clear rTable;
                rTable.hopDistance = INFINITY;
                canSleep = true;
                start goToSleepTimer;
            ENDIF
            IF p.hopCount > rTable.hopDistance THEN
                drop packet p;
                BREAK;
            ENDIF
            IF p.hopCount < rTable.hopDistance THEN
                clear rTable;
                rTable.hopDistance = INFINITY;
                forwardFlag = true;
            ENDIF
            rTable.hopDistance = p.hopCount;
            add new routing entry to rtable;
            p.hopcount = p.hopcount + 1;
            IF p.hopcount < TTL and forwardFlag == true THEN
                forward packet p;
            ENDIF
            BREAK;
        CASE TRAIL_BEACON:
            IF p.seqNO == trailSeqNo THEN
                drop packet p;
                                                (to be continued)
```

```
        ELSE
            clear rTable;
            trailSeqNo = p.seqNO;
            rTable.hopDistance = p.hopCount;
            add new routing entry to rtable;
            p.hopcount = p.hopcount + 1;
            IF p.hopcount < trailTTL THEN
            forward packet p;
            ENDIF
        ENDIFELSE
        BREAK;
    CASE DATA:
        IF p.destNode == THIS_NODE THEN
            forward packet p;
            BREAK;
        ELSE IF p.srcNode == rTable.nextHop THEN
            dataForwarded = true;
        ENDIFELSE
        drop packet p;
        BREAK;
    CASE NO_SLEEP:
        canSleep = false;
        BREAK;
    CASE HELP:
        IF has route to a mobile sink and
                  rTable.nextHop != p.srcNode THEN
              send back a HELP_RESPONSE packet;
        ENDIF
        BREAK;
    CASE HELP_RESPONSE:
        IF helpRequested == true THEN
            clear rTable;
                                          (to be continued)
```

```
                rTable.hopDistance = p.hopCount;
                add new routing entry to rtable;
                helpRequested = false;
            ENDIF
            BREAK;
    ENDSWITCH
ENDIF

IF routing path breaks THEN
    broadcast a HELP message to one-hop neighbors;
ENDIF

IF data is generated THEN
    destNode = rTable.nextHop;
    send DATA packets;
ENDIF

IF multiple next hop candidates are found THEN
    destNode = rTable.nodeWithMostEnergy;
    send NOSLEEP packet;
ENDIF

IF goToSleepTimer expires THEN
    IF canSleep == true THEN
        turn off radio and go to sleep;
        start sleepTimer;
    ENDIF
ENDIF

IF sleepTimer expires THEN
    wake up and turn on radio;
ENDIF
END WHILE
```

Figure 3.8: Pseudo code of the sensor node in E-Trail

8. When an event is detected nearby and corresponding data has been generated, the node becomes a source and sends data to its next hop.

9. When the sensor node has collected enough information about its next hop candidates during the cluster formation phase, it selects a next hop candidate with the most remaining energy and sends a NO_SLEEP message to that node.

The corresponding algorithms of the sensor node in E-Trail are presented in Figure. 3.8.

3.4 Summary

In this chapter, we have explained our proposed protocol — the Energy efficient TRAIL-based data dissemination protocol (E-Trail) in terms of the motivation, the important technical features and the design and implementations. E-Trail employs the combination of a trail-based approach with a sleep-wake mechanism. To our best knowledge, it is the first flat data dissemination protocol for WSNs with mobile sinks that allows redundant sensors in the network to sleep for a certain period of time. Under the premise of fast, reliable and effective data delivery within the sensor network, E-Trail further reduces the average energy consumption and improves the network lifetime by exploring the energy consumed during the idling state of sensors.

Chapter 4

Performance Evaluation

To evaluate the performance of E-Trail, experiments have been conducted to compare it with four other flat data dissemination protocols (Dual-Sink, Efficient Routing, LURP and ALURP) reviewed in Chapter 2. Furthermore, besides the original E-Trail protocol, we also implement a version of E-Trail in which the sleeping mechanism is turned off. The protocols are evaluated through different sets of simulations using the network simulator ns-2 [2]. The simulation results confirm the energy efficiency and reliability of E-Trail to deliver data from multiple sources to multiple mobile sinks.

4.1 Methodology and Metrics

Due to the lack of hardware devices such as wireless sensors and mobile sinks, the experiments cannot be carried out in a real testing environment. Therefore, we use ns-2 to simulate a wireless sensor network environment, where the sources, sensors and sinks of the data dissemination protocols are implemented as agents. The agents can capture the essence of the protocols, *e.g.*, the cluster formation, the sleep-wake mechanism, the trail generation and the path recovery of E-Trail. The E-Trail protocol is implemented as two types of agents: the $TrailSinkAgent$, which is carried by a moving node and functions as a mobile sink traveling in the sensor field to continuously collect sensor data, and the $TrailSensorAgent$, which is set on a stationary node and functions as a sensor deployed in the network that sometimes can become a data source upon the detection of certain events. The corresponding algorithms of the two agents are presented in Figure. 3.5 and Figure. 3.8, respectively. The other compared protocols such as Dual-Sink, Efficient Routing, LURP, ALURP and E-Trail without sleeping, are all implemented in a similar

way.

A series of simulations have been run for each protocol through the ns-2 to evaluate their network performances, including the data delay, the energy consumption, the delivery rate, the control overhead and the network lifetime. The data delay is the average time between the moment a source transmits a data packet and the moment a sink receives the packet. This metric indicates the timeliness of the data packets in the protocols. The energy consumption is the average amount of sensor energy consumed by each node in the network. The communication (transmitting and receiving) energy and the idle energy are both counted in our simulations; however, to have a clearer comparison of the energy efficiency of the routing algorithms used in different protocols, we also have a set of simulations that excludes the idle energy consumption. The delivery rate is the ratio of the number of successfully received data packets to the number of all the data packets generated and sent by the sources. The delivery rate shows the reliability of the data dissemination process. The control overhead is the number of packets sent and forwarded by all the sensors in the network to control and assist the data delivery to the mobile sinks, *i.e.*, the number of all the packets other than data that are transmitted during the network operation. As for the network lifetime, several definitions can be found in the literature. Some consider the network lifetime as the time when the first node in the network runs out of energy, whereas others prefer the time corresponding to the last data packet received by the sink. In our work, we use the number of nodes alive in the sensor network coordinated with the time line to present the network lifetime in a more straightforward way.

4.2 Simulation Model

The protocols are evaluated with the same network configurations and simulation scenarios. We use IEEE 802.11 as the underlying MAC protocol. The wireless bandwidth is 38.4 Kbps. The transmitting, receiving and idling power consumption rates of the sensor node are set to 0.30 W, 0.30 W and 0.209 W respectively, based on the iMote2 board specifications [3]. Specifically for E-Trail, the sleeping power rate is 0.002 W, and the transit power to switch between the active mode and the sleeping mode for the sensors is 0.0001 W. We assume the initial energy of the sensor nodes are enough to complete the simulations, except in the network lifetime evaluations. The radio transmission range of all the sensors and the mobile sinks are set to 100m. The overview of the network parameters in our simulations can be found in Table. 4.1, and the corresponding Tcl

code to configure these parameters in the ns-2 is presented in Figure. 4.1.

Parameter	**Value**
Radio Range	100m
TX Power	0.30W
RX Power	0.30W
Idle Power	0.209W
Max Sink Speed	20 m/s
Data Interval	1 sec
Round Period	50 sec
Simulation Time	1000 sec
Packet Size	44 bytes
Bandwidth	38.4 Kbps
Number of Nodes	1024
Number of Sinks	2 - 12
Number of Sources	4, 10
Deployment Area	2000x2000m

Table 4.1: Overview of the simulation parameters

The default simulation scenario has 1024 sensor nodes deployed across a 2000m x 2000m field. Nodes are distributed following a disturbed grid topology, *i.e.*, the x and y coordinates of each node are displaced by a random distance from its position on the grid in order to avoid large differences in the node density. The maximum distance between two neighboring nodes is 60m. The source nodes are randomly selected from the sensor field every 50 seconds, and each source generates one data packet per second. In case the number of sources has a great impact on the network performances, we run every simulation set with both 4 sources and 10 sources. The initial positions of the mobile sinks are randomly generated for each execution. The mobile sinks move according to the standard random way-point model with a maximum speed of 20m/s. Each simulation run lasts for 1000 seconds, and the result is plotted with a 95% confidence interval from 10 simulation runs.

There are some protocol-specific parameters, such as the radius of the destination area in LURP and ALURP. Since the descriptions in the original literature did not state how to decide the proper values for these parameters clearly, the rule of thumb to deal with them in our simulations is to use the value which leads to the best simulation

performance. Respectively, the mobile sink of Dual-Sink sends out HELLO messages every 20 seconds, and the TTL of the mobile sink HELLO messages is 5; the cost change threshold in Efficient Routing is 60%; the radius of the destination area in both LURP and ALURP is 150m; the trail beacon rate of the mobile sinks in E-Trail is every 2 seconds.

```
set val(chan)       Channel/WirelessChannel    # channel type
set val(prop)       Propagation/TwoRayGround   # radio-propagation model
set val(netif)      Phy/WirelessPhy            # network interface type
set val(mac)        Mac/802_11                 # MAC type
set val(ifq)        Queue/DropTail/PriQueue    # interface queue type
set val(ll)         LL                         # link layer type
set val(ant)        Antenna/OmniAntenna        # antenna model
set val(ifqlen)     50                         # max packet in ifq

set val(engmodel)   EnergyModel
set val(txPower)    0.30                       # transmitting energy
set val(rxPower)    0.30                       # receiving energy
set val(idlePower)  0.209                      # idling energy
set val(initeng)    1000                       # Initial energy
set val(rxthresh_)  1.51875e-08                # for 100 meters

set val(nn)         1024                       # number of mobile nodes
set val(nxn)        32                         # matrix of nxn nodes
set val(dist)       60                         # max distance between 2
                                               # nodes
Mac/802_11 set dataRate_    39321.6            # 38.4 Kbps
```

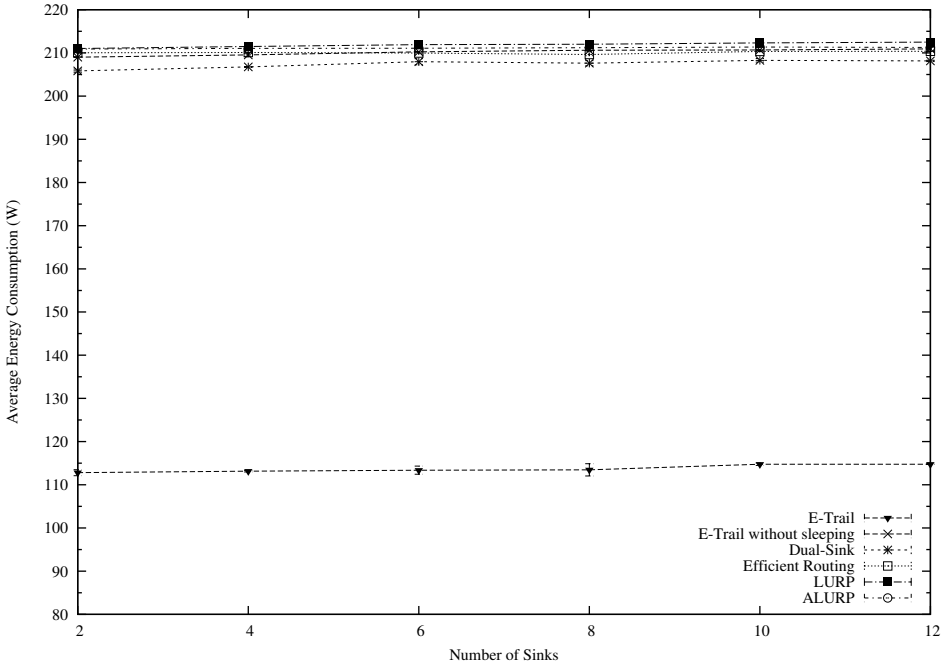
Figure 4.1: The Tcl code for setting the network parameters in ns-2

4.3 Simulation Results

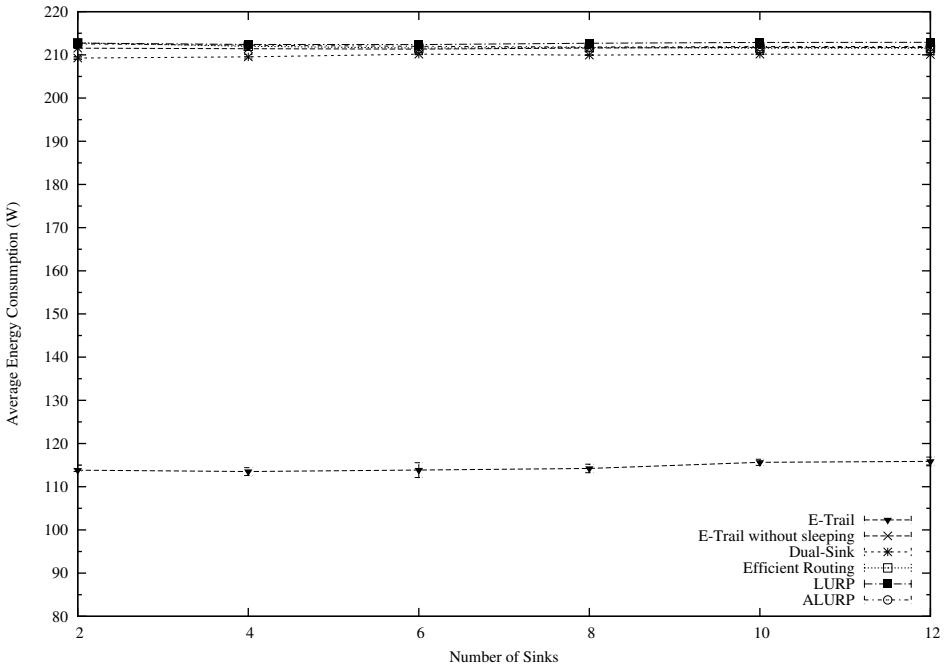
4.3.1 Energy Consumption

In the first group of experiments we evaluate the average energy consumption of E-Trail and the other protocols. In Figure. 4.2 we can observe that the number of sources and the number of mobile sinks have little or no impact on the energy consumption. Moreover, the results show that the proposed E-Trail protocol can save up to 50% sensor energy compared to the other selected protocols (in E-Trail each node consumes about 113 W of sensor energy in average during the 1000 seconds of simulation time, whereas in E-Trail without sleeping, Dual-Sink, Efficient Routing, LURP and ALURP, each node consumes around 210 W of sensor energy in average). We conclude that the sleeping mechanism of E-Trail is the major reason for energy saving, as evidenced by the significant difference among the protocols investigated. However, a curious observation is the energy consumption of the other five protocols, including E-Trail without sleeping. We believe this analogous performance is due to the fact that sensor nodes spend more time in an idling state than in any other state, thereby indicating that the energy consumption in the idling state dominates the results of the five schemes.

In order to have a clearer view of the energy efficiency of the routing algorithms in the protocols, we set the power consumption rate of the idling state to 0, carried out another experiment with 10 sources, and evaluated the sensor energy consumed only by routing updates and data delivery. The result is presented in Figure. 4.3. We can observe again that E-Trail consumes less energy than the majority of the evaluated protocols because it utilizes the trail generation technique to minimize the overhead of sink location updates. However, E-Trail without sleeping, which employs a same trail-based routing update algorithm, does not seem to have a competitive performance. This is because the sensor nodes in E-Trail switch to sleep mode if they do not receive NO_SLEEP messages. When a node is sleeping, its radio is turned off and it cannot receive any packets. Therefore, when the mobile sink periodically broadcasts the trail beacons to the nearby sensor field, much fewer nodes will receive and forward the trail beacons in E-Trail than in E-Trail without sleeping. As a result, less sensor energy is consumed by the packet transmitting in E-Trail. Note that none of the evaluated protocols has an average energy consumption of more than 16 W when the idling energy consumption rate is 0. If we compare this to the previous simulations that take idle energy into account, more than 90% of the overall sensor energy is consumed during the idling state. Although this ratio could vary



(a)



(b)

Figure 4.2: The energy consumption experimental results. (a) The average energy consumption with 4 sources. (b) The average energy consumption with 10 sources.

with different network configurations, it still proves the demand for data dissemination protocols like E-Trail which attempt to reduce idling energy consumption.

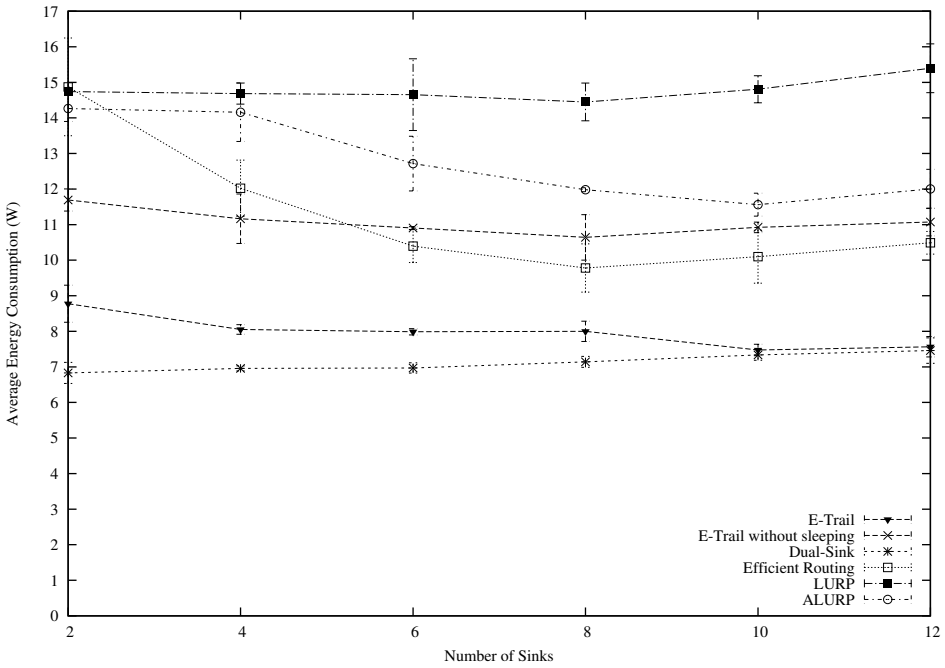
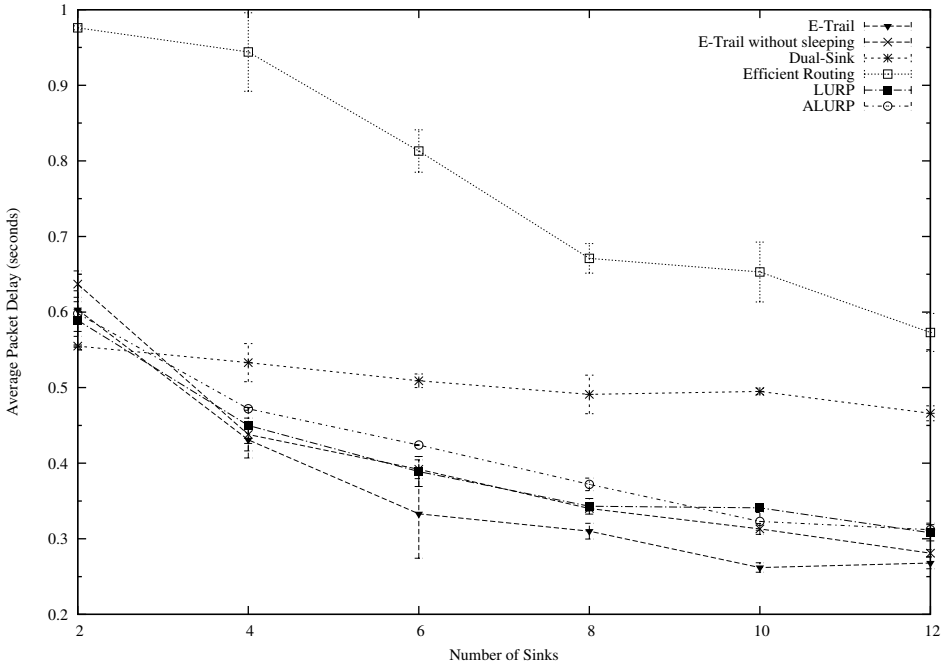


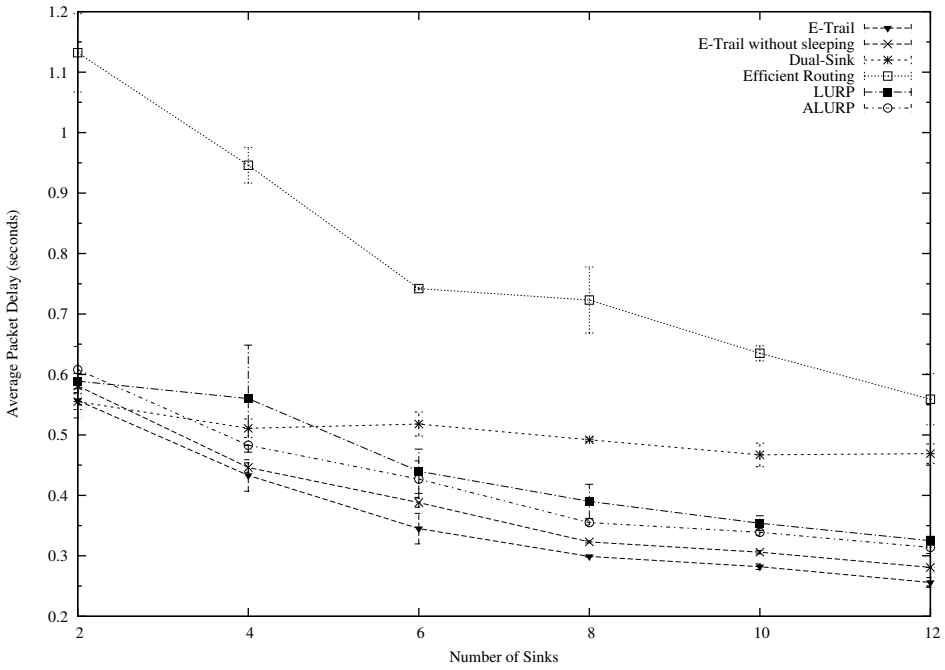
Figure 4.3: The energy consumption experimental result with 10 sources when the idling power rate is set to 0.

4.3.2 Data Delay

In the second set of simulations, we investigated the average data delay of E-Trail and the other protocols. As we can tell from Figure. 4.4, the number of sources does not have a significant impact on the average data delay. The data delay of E-Trail, E-Trail without sleeping, Efficient Routing, LURP and ALURP all have clear trends to reduce as the number of mobile sinks increases. Such situation makes perfect sense, as all the above protocols are essentially flooding based protocols, where the sensors have a tendency to select the nearest mobile sink as the destination to which they direct data. Therefore, as more mobile sinks are deployed in the sensor network, the average length of routing paths is almost certainly shortened, resulting in a reduced average data delay. Although a similar trend can be found in the Dual-Sink protocol, the data delay of Dual-Sink seems not to be as dramatically affected by the number of mobile sinks as in the other



(a)



(b)

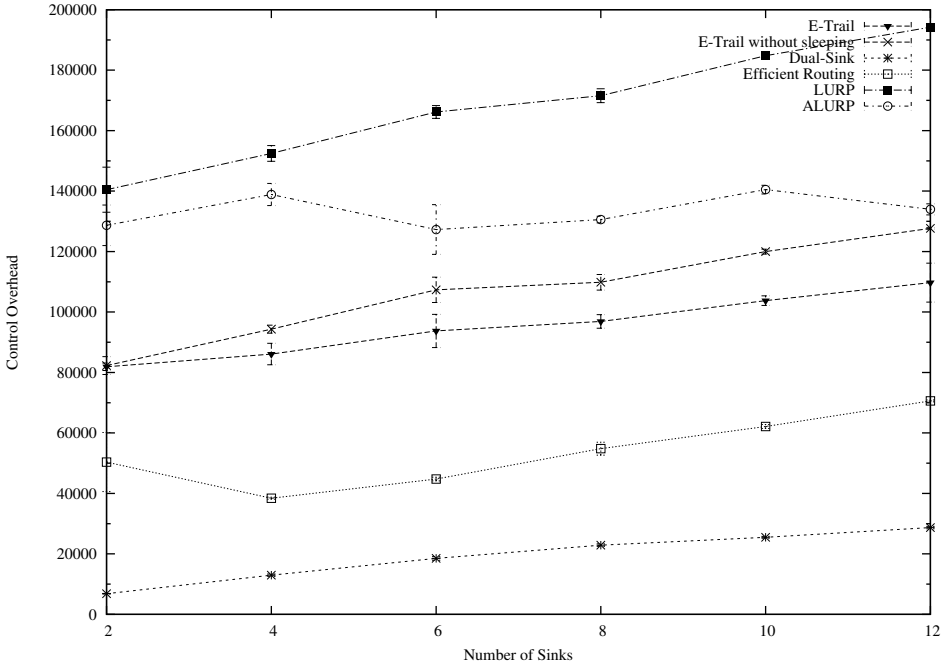
Figure 4.4: The data delay experimental results. (a) The data delay with 4 sources. (b) The data delay with 10 sources.

protocols. This is mainly because the mobile sink of Dual-Sink only collects data from the sensor nodes that are less than several hops away at one time; thus the influence of a mobile sink can be very limited from the perspective of the entire network, especially when the network is very large.

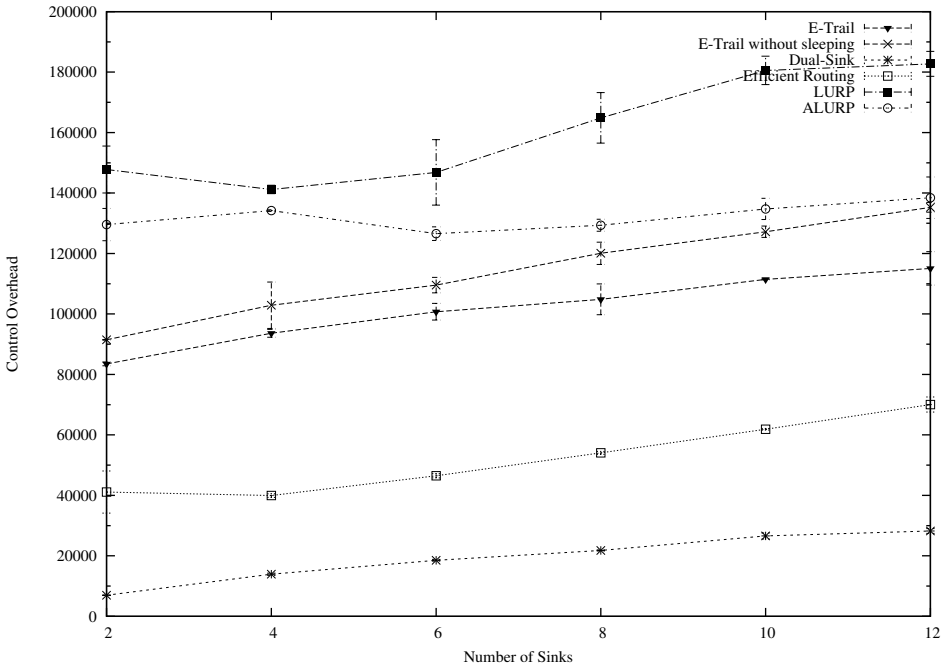
When it comes to the individual performance of each protocol, E-Trail has the shortest data delay among all the simulated protocols. It actually performed a little beyond our expectations, since the routing path of E-Trail generally follows the movement trail of the mobile sink and therefore is not optimal. We can only assume with a round duration of as short as 50 seconds, E-Trail manages to reset all the routing paths and restart the trail generation process before the trail paths detour too far away from the optimal routes. On the contrary, the Efficient Routing protocol has the longest data delay. It is not difficult to understand because the rationale behind the protocol is that since the nodes farther away from the mobile sink send their data through many hops, a slightly higher forwarding cost along the route does not cause a great performance degradation compared to the energy gain from using much fewer route updates. In other words, the protocol is saving sensor energy at the cost of the prolonged routing paths and the extended data delay. As the sensors in Efficient Routing only forward the sink beacons when the change in cost reaches a threshold, we can usually find the data forwarding path of Efficient Routing to detour quite far away from the ideal shortest path during the simulations, which inevitably increases the time of data forwarding. As for the other protocols, E-Trail without sleeping, LURP and ALURP have excellent data delay performance similar to E-Trail; Dual-Sink provides fairly good results, but not as good as the other protocols except Efficient Routing, since a lot of data packets in Dual-Sink are forwarded to the static sink directly when there is not a mobile sink nearby.

4.3.3 Overhead

In the third simulation set, we study the control overhead of the evaluated protocols. As the number of sources as well as the data generation intervals of the protocols are the same, the results of the overhead shown in Figure. 4.5 reflect the number of control packets used for building the routes, updating the locations of the mobile sinks, maintaining the data forwarding paths and coordinating the data delivery, so as to forward the same number of data packets to the mobile sinks in the protocols. Since the flat data dissemination protocols are usually not source-oriented, the overhead of all the six protocols are not affected much by the number of sources. However, the number of mobile sinks has a



(a)



(b)

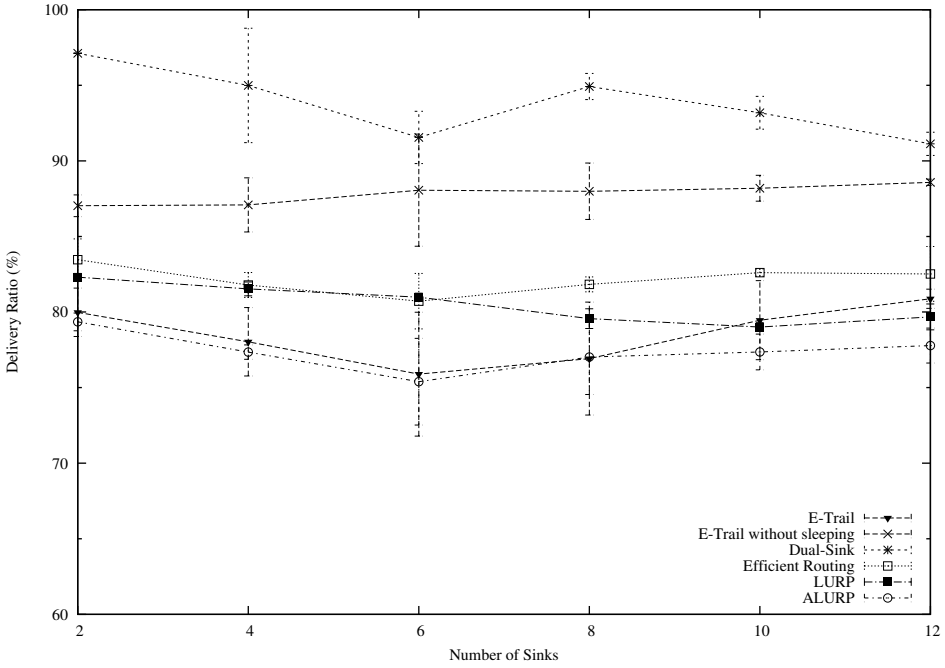
Figure 4.5: The overhead experimental results. (a) The overhead with 4 sources. (b) The overhead with 10 sources.

direct impact on the overhead. This is quite straightforward because more mobile sinks means more location updates, which in many cases is the only type of network overhead in flat protocols.

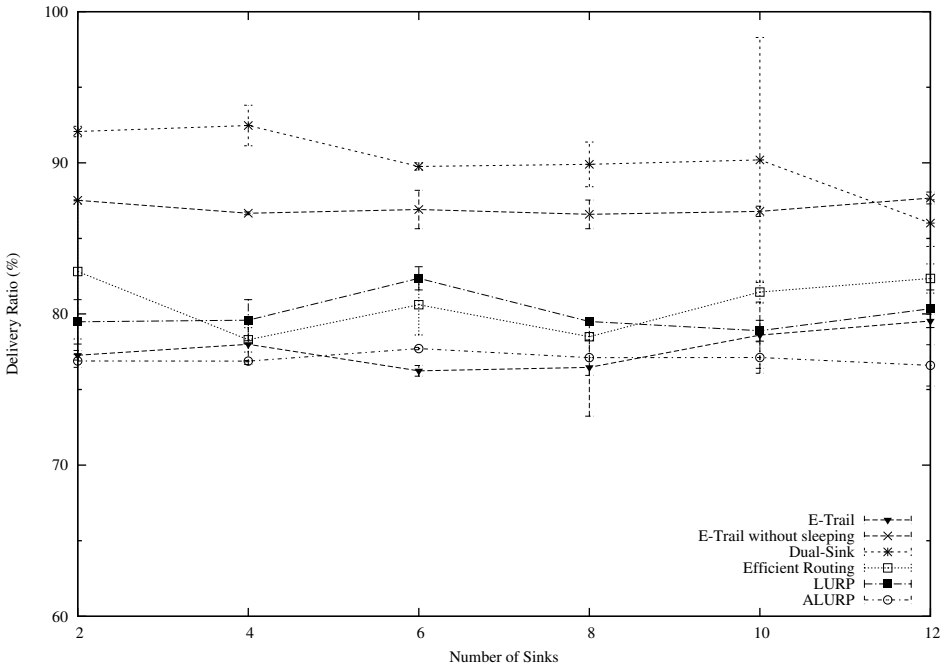
Among the six compared protocols, LURP has the highest overhead, which is more than 140,000 control packets during the 1000 seconds of simulation; ALURP has a slightly lower overhead than LURP, since the major amelioration of ALURP from LURP is to reduce the size of the local area that the mobile sink broadcasts its location information to; E-Trail without sleeping and E-Trail follow, because unlike LURP and ALURP, the mobile sinks in E-Trail without sleeping and E-Trail only broadcast the trail beacons to the sensor nodes less than two hops away. Moreover, the trail-based update mechanism does not flood the sensor field, whereas in LURP and ALURP, as soon as the mobile sink moves out of the destination area, it has to broadcast to the entire network to build a new destination area. The Efficient Routing protocol manages the overhead very well by setting a cost change threshold at each sensor node; however, as we have mentioned, the data delay is then inevitably extended. Dual-Sink has the lowest overhead among all the evaluated protocols. It employs a static sink that only broadcasts its location to the entire network once at the very beginning of the network operation. Then, the mobile sink of Dual-Sink only plays a role of alleviating the hot spot problem. The mobile sink is not always connectable to the sensor network, and it only sends out beacons to the nearby sensors once in a while to collect data from them. In summary, of all the pure mobile sink approaches investigated here, which exclude the Dual-Sink protocol, the overhead performance of E-Trail is among the most competitive ones. Knowing that Efficient Routing is sacrificing its average data delay in order to reduce the control overhead, E-Trail has achieved a perfect balance between the two important network metrics.

4.3.4 Delivery Rate

Figure. 4.6 depicts the results of delivery rate. It can be observed that E-Trail maintains a stable performance around 80%, which is comparable to Efficient Routing, LURP and ALURP. However, when compared to Dual-Sink and E-Trail without sleeping, we conclude that the sleeping mechanism of E-Trail affects the data delivery rate due to the loss of the NO_SLEEP messages. When the sensor node in E-Trail sends a NO_SLEEP message to one of its next hop candidates, it assumes the destination node of this message will be awake during the rest of the current round. Thus, when the NO_SLEEP message



(a)



(b)

Figure 4.6: The delivery rate experimental results. (a) The delivery rate with 4 sources. (b) The delivery rate with 10 sources.

is lost, the node which is supposed to be awake goes to sleep, and consequently a series of consecutive data packets will be lost until the path recovery takes place and finds an alternate route.

4.3.5 Network Lifetime

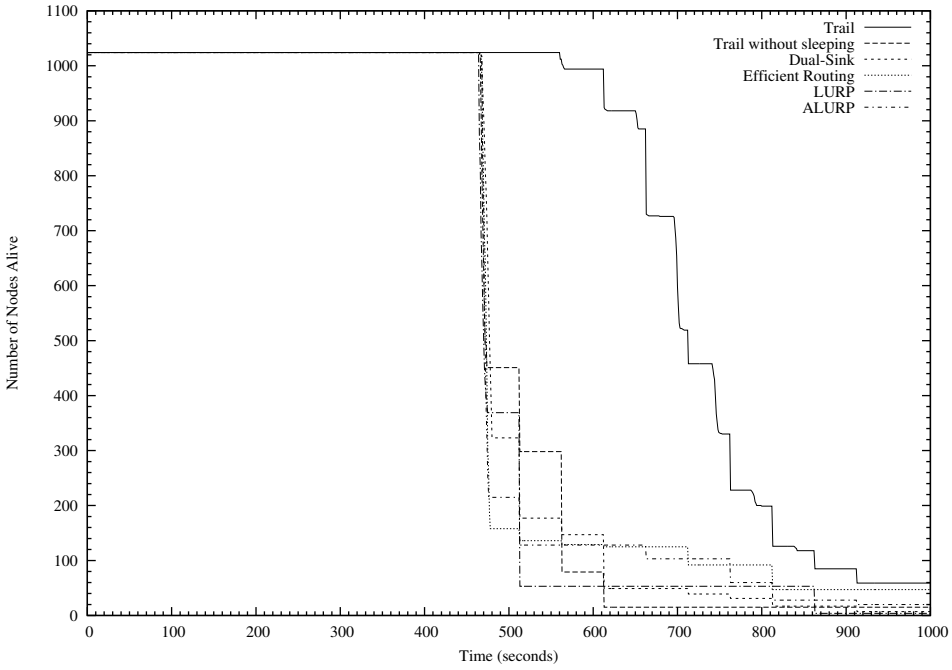


Figure 4.7: The network lifetime experimental result with 10 sources and 6 sinks. The initial energy of each sensor is set to 100 W.

In the last set of simulations, the network lifetime performance of each protocol is evaluated by giving the sensor nodes low initial energy so that the majority of the sensors will use up their energy during the 1000 seconds of simulation time. In this thesis, we consider the network lifetime as the time spot when the first node runs out of energy in the network. To this end, the initial energy of each sensor is set to 100 W, 6 mobile sinks are randomly deployed in the sensor field and 10 nodes are randomly selected as the data sources. As depicted in Figure. 4.7, E-Trail improves the network lifetime significantly because of its sleep-wake mechanism. The simulation result shows an improvement of at least 20%. Furthermore, if we take the network with around 80% to 90% nodes alive into account, which is still able to operate normally without losing many data packets,

the network lifetime performance of E-Trail can be considered even better (up to 45% longer than the other protocols). The declining curve of the number of nodes alive in the network of E-Trail shown in Figure. 4.7 is much gentler and smoother than the other compared protocols. Again, the network lifetime result proves the sensor energy consumed during the idling state dominates the network lifetime of each protocol, as we find that the first nodes in different protocols (except for E-Trail) use up their energy at almost the same time.

4.4 Summary

Five sets of simulations in total are presented in this chapter to evaluate the performance of the proposed E-Trail protocol. The results show that among the evaluated flat data dissemination protocols for WSNs with mobile sinks, E-Trail provides the best energy efficiency and the longest network lifetime, while its data delay and control overhead performances are also maintained at very high levels. The delivery rate of E-Trail is not among the best ones due to the sleep-wake mechanism, but it is still acceptable and comparable to the majority of the other selected protocols. The sleeping mechanism of E-Trail is the main reason for its great energy performance, because it not only saves sensor energy by putting a lot of sensor nodes into sleep mode, but also reduces the overhead of sink location updates with less awake sensors in the sensor field receiving and forwarding the trail beacons from the mobile sinks. In summary, E-Trail is a fast, reliable and energy efficient data dissemination protocol to support multiple mobile sinks in the WSNs, and also a pioneer in the related research areas to explore the idling state energy consumption of sensor nodes.

Chapter 5

Conclusion and Future Work

Various protocols have been proposed in recent years to exploit the energy efficiency in data dissemination processes with mobile sinks in WSNs, but almost all of them have ignored the fact that energy consumed when nodes fall into idle state dominates the overall energy consumption. Inspired by this, a new data dissemination protocol E-Trail has been proposed in this thesis to explore the idling state energy consumption of sensors and provide reliable, energy efficient data dissemination over WSNs with mobile sinks. In this chapter, a summary of our thesis work will be presented and the potential for future research will be discussed.

5.1 Conclusion

In this thesis, the discussion of relevant research works has been presented in Chapter 2 in terms of the classification of the data dissemination protocols for mobile sinks and the review of major mobile sink protocols. The mobile sink protocols can be classified in many different ways based on their characteristics and features, such as the application type, the data collection scheme, the disseminated information type, the network architecture, and the sink mobility pattern. In the later part of Chapter 2, we described and investigated a number of currently existing data dissemination protocols for WSNs with mobile sinks. Since the hot spot problem that usually exists in the static sink approaches has been eliminated by the sink mobility, the mobile sink protocols mainly focus on how to efficiently track the movement of the sink and how to reduce the overhead and energy consumption of the sink location updates. The Dual-Sink protocol, the Efficient Routing protocol, the LURP protocol and the ALURP protocol, which we call the flat data

dissemination protocols, are based on frequent broadcasting initiated by the mobile sink to update its location to the sensor network. The above protocols all successfully constrain the sink location updates among a very limited number of sensors in the network, rather than through the entire sensor field. The TTDD protocol, the LBDD protocol, the Railroad protocol and the CODE protocol, which we call the virtual infrastructure based protocols, adopt another type of method to reduce the overhead by organizing the physical network topology into overlay virtual structures such as grids, strips, circles or clusters. Then, the sink queries and the sensor data can be forwarded over these virtual structures only, without the participation of the other nodes in the sensor network that do not belong to the virtual structures. The virtual infrastructure also enables these types of protocols to scale with very large networks. Although the protocols investigated in Chapter 2 all strive to minimize the control overhead and maximize the energy efficiency, almost all of them, except for CODE, have ignored the fact that the idling energy consumption of sensor nodes dominates the overall energy consumption. Without paying attention to the energy consumed during the idling state of sensors, none of these protocols have achieved optimal energy efficiency. However, even though the CODE protocol has introduced a novel sleep-wake mechanism to reduce the idling energy consumption, the technique to equivalently partition the network into virtual grids in CODE is not practical in real world applications, due to a variety of reasons. Furthermore, CODE is a virtual infrastructure-based protocol, which is not quite adequate to be applied to the small or medium sensor networks, according to our investigations. Therefore, new flat data dissemination protocols should be developed to provide more practical mechanisms to reduce the energy consumption of the idling state, and to support fast, reliable and energy efficient data delivery over WSNs with mobile sinks.

Motivated by the strengths and weaknesses of current data dissemination protocols, we propose a new Energy efficient TRAIL-based data dissemination protocol — E-Trail, which is a combination of a trail-based approach with clustering and a sleep-wake mechanism. In Chapter 3 we extensively discussed the motivation of proposing E-Trail, and its important technical features, including the cluster formation, the sleep-wake mechanism, the trail generation, and the path recovery. The cluster formation is inherited from E-Trail's precursor the MDC/PEQ protocol [38], where the mobile sink uses restricted beacon broadcasting with a specific TTL to build a cluster periodically. The sleep-wake mechanism lets every single node in the cluster sleep unless some sensor node tells it not to. The trail generation is inspired by the image of a marked or beaten path left by a human body's feet as he or she walks through the woods or wilderness. The

path recovery phase is triggered when the routing path fails or breaks. The design and implementation of these E-Trail features have been further addressed in the rest part of Chapter 3. Since there are two types of devices participating in the data dissemination process of E-Trail, namely the mobile sink and the sensor node, we have depicted the work flow and the corresponding algorithms for each of them separately.

Extensive experiments have been carried out through the network simulator ns-2 to evaluate the performance of the proposed E-Trail protocol. Different sets of simulations have been executed to evaluate the energy consumption, the data delay, the control overhead, the delivery rate and the network lifetime of E-Trail compared to the other selected protocols. The simulation results are presented and analyzed in Chapter 4. The results have shown that E-Trail saves up to 50% sensor energy and significantly improves the network lifetime, while the performance of E-Trail regarding other network metrics such as the data delay, the control overhead and the delivery rate are also considered as excellent, or at least comparable to the other protocols. The simulation results prove the success of E-Trail and confirm the urgent need for designing mobile sink protocols for WSNs that exploit the idling energy consumption of sensor nodes to achieve optimal energy efficiency.

5.2 Future Work

Some subsequent areas for future research into the possibilities offered by adopting E-Trail are as follows:

- *Deal with the NO_SLEEP message lost.* The unique sleep-wake mechanism of E-Trail relies on the successful delivery of the NO_SLEEP messages. When a NO_SLEEP message gets lost, the sensor node that is supposed to be awake for the rest of the current round will go to sleep. Therefore, consecutive data packets will be lost before the node which sent the NO_SLEEP message eventually realizes its next hop has switched to the sleep mode and then triggers the path recovery mechanism. Currently we do not have effective solutions to prevent the NO_SLEEP messages from getting lost. The research of this particular issue is still ongoing.

- *The relationship between the sink speed and the trail beacon rate.* None of the current mobile sink protocols based on broadcasting has come up with a clear method (or a formula) to determine the frequency of the mobile sink beacons. We realize this frequency should be directly affected by the sink speed and adjusted

dynamically; however, we have not yet proposed a formal method that can represent this relationship. As most of the current protocols do, we choose the trail beacon rate of E-Trail quite arbitrarily and subjectively. Still, further research on the relationship between the sink speed and the frequency of location updates should be carried out, because once this problem is perfectly solved, the delivery rate and the energy efficiency of similar mobile sink approaches would be dramatically improved.

- *New sleep − wake mechanism.* The E-Trail protocol does not require its sensor nodes to be location-aware. Thus, the sleep-wake mechanism of E-Trail is completely based on the delivery of the NO_SLEEP messages. However, as we have already discussed, this scheme has a few drawbacks. We believe by incorporating the location-aware sensors into E-Trail, we will be able to propose some better sleep-wake mechanisms that utilize the geographical information of the sensor nodes as well to determine which nodes should go to sleep and which nodes should be awake.

Appendix A

Glossary of Terms

ACK Acknowledgement.

ALURP The Adaptive Location Update-based Routing Protocol.

CFG Configuration.

CODE The COordination-based data Dissemination protocol for wireless sEnsor networks.

Dual-Sink The Dual-Sink protocol.

Efficient Routing The Efficient Routing protocol.

E-Trail The Energy efficient TRAIL-based data dissemination protocol.

GPS Global Positioning System.

Kbps Kilobyte per second.

LBDD The Line-Based Data Dissemination protocol.

LURP The Local Update-based Routing Protocol.

ns-2 The network simulator ns-2.

PMDD The Predictable Mobility-based Data Dissemination protocol.

QoS Quality of Service.

sec Second(s).

Tcl Tool command language, a scripting language.

TTDD The Two-Tier Data Dissemination protocol.

TTL Time to Live.

W Watt.

WSNs Wireless Sensor Networks.

Bibliography

[1] http://en.wikipedia.org/wiki/wireless_sensor_network/.

[2] http://isi.edu/nsnam/ns/.

[3] Memsic iMote2 IPR2400 Processing and Sensing Board - Data Sheet, 2010.

[4] S. Ahn and D. Kim. Proactive context-aware sensor networks. In *Proceedings of European Workshop on Wireless Sensor Networks (EWSN)*, pages 38–53, feb. 2006.

[5] H.M. Ammari and S.K. Das. An energy-efficient data dissemination protocol for wireless sensor networks. In *Pervasive Computing and Communications Workshops, 2006. PerCom Workshops 2006. Fourth Annual IEEE International Conference on*, pages 5 pp. –363, mar. 2006.

[6] A. P. Azad and A. Chockalingam. Mobile base stations placement and energy aware routing in wireless sensor networks. In *Proc. of IEEE WCNC06*, volume 1, pages 264–269, apr. 2006.

[7] S. Basagni, A. Carosi, E. Melachrinoudis, C. Petrioli, and Z.M. Wang. Controlled sink mobility for prolonging wireless sensor networks lifetime. *Wireless Networks*, 14(6), dec.

[8] Y. Bi, L. Sun, J. Ma, L. Na, I.A. Khan, and C. Chen. Hums: An autonomous moving strategy for mobile sinks in data-gathering sensor network. *EURASIP Journal on Wireless Communications and Networking*, 2007.

[9] D. Braginsky and D. Estrin. Rumor routing algorithm for sensor networks. In *Proceedings of the First Workshop on Sensor Networks and Applications (WSNA)*, pages 22–31, oct. 2002.

[10] Arnab Chakrabarti, Ashutosh Sabharwal, and Behnaam Aazhang. Using predictable observer mobility for power efficient design of sensor networks. In *Proc. of the 2nd intl. conf. on Information processing in sensor networks, IPSN'03*, pages 129–145, 2003.

[11] I. Chatzigiannakis and A. Kinalis. Fast and energy efficient sensor data collection by multiple mobile sinks. In *Proceedings of the 5th ACM international workshop on Mobility management and wireless access (MOBIWAC)*.

[12] I. Chatzigiannakis, A. Kinalis, and S. Nikoletseas. Sink mobility protocols for data collection in wireless sensor networks. In *Proceedings of the 4th ACM international workshop on Mobility management and wireless access*, pages 52–59, oct. 2006.

[13] I. Chatzigiannakis, A. Kinalis, and S. Nikoletseas. Efficient data propagation strategies in wireless sensor networks using a single mobile sink. *Computer Communications Journal*, 31, 2008.

[14] K. Dantu, M. Rahimi, H. Shah, S. Babel, A. Dhariwal, and G.S. Sukhatme. Robomote: enabling mobility in sensor networks. *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 404 – 409, apr. 2005.

[15] Kristóf Fodor and Attila Vidács. Efficient routing to mobile sinks in wireless sensor networks. In *Proc. of the 3rd Intl. Conf. on Wireless internet, WICON'07*, pages 1–7, ICST, Brussels, Belgium, 2007.

[16] S.R. Gandham, M. Dawande, R. Prakash, and S. Venkatesan. Energy efficient schemes for wireless sensor networks with multiple mobile base stations. *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE*, 1:377 – 381 Vol.1, dec. 2003.

[17] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(4):11–25, oct. 2001.

[18] E. B. Hamida and G. Chelius. A line-based data dissemination protocol for wireless sensor networks with mobile sink. In *Proc. of the 2008 IEEE International Conference on Communications ICC*, pages 2201–2205, may. 2008.

[19] E.B. Hamida and G. Chelius. Strategies for data dissemination to mobile sinks in wireless sensor networks. *Wireless Communications, IEEE*, 15(6):31–37, dec. 2008.

[20] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS '00)*, page 10 pp. vol.2.

[21] W. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proc. 5th ACM/IEEE Mobicom Conference (MobiCom '99)*, pages 174–185, aug. 1999.

[22] K. Hwang and D. Eom. Adaptive sink mobility management scheme for wireless sensor networks. *Ubiquitous Intelligence and Computing*, 4159:478–487, 2006.

[23] K. Hwang, J. In, and D. Eom. Distributed dynamic shared tree for minimum energy data aggregation of multiple mobile sinks in wireless sensor networks. *Lecture Notes in Computer Science*, 3868:132–147, 2006.

[24] H. Jeon, K. Park, D.J. Hwang, and H. Choo. Sink-oriented dynamic location service protocol for mobile sinks with an energy efficient grid-based approach. *Sensors*, 9(3):1433–1453, 2009.

[25] B. Karp and H.T. Kung. Greedy perimeter stateless routing for wireless networks. In *ACM/IEEE International In Conference on Mobile Computing and Networking (MOBICOM)*, pages 243–254, aug. 2000.

[26] N. M. Khan and I. Ali. Quasi centralized clustering approach for an energy-efficient and vulnerability-aware routing in wireless sensor networks. In *Proceeding of the 1st ACM international workshop on Heterogeneous sensor and actor networks*, pages 67–72, 2008.

[27] H. S. Kim, T.F. Abdelzaher, and W.H. Kwon. Minimum-energy asynchronous dissemination to mobile sinks in wireless sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems (SenSys03)*, pages 193–204, 2003.

[28] K. Kweon, H. Ghim, Hong J., and H. Yoon. Gradient-based energy-efficient routing from multiple sources to multiple mobile sinks in wireless sensor networks. In *4th International Symposium on Wireless Pervasive Computing (ISWPC)*, 2009.

[29] Euisin Lee, Soochang Park, Fucai Yu, Younghwan Choi, Min-Sook Jin, and Sang-Ha Kim. A predictable mobility-based data dissemination protocol for wireless sensor networks. In *Proceedings of the 22nd International Conference on Advanced Information Networking and Applications*, pages 741–747, 2008.

[30] C.J. Lin, P.L. Chou, and C.F. Chou. Hcdd: Hierarchical cluster-based data dissemination in wireless sensor networks with mobile sink. In *Proc. 2006 Intl. Conf. Wireless Commun. and Mobile Comp.*, pages 1189–1194, 2006.

[31] Haiyun Luo, Fan Ye, Jerry Cheng, Songwu Lu, and Lixia Zhang. TTDD: Two-tier data dissemination in large-scale wireless sensor networks. *Wireless Networks*, 11(1-2):161–175, 2005.

[32] J. Luo and J. Panchard. Mobiroute: routing towards a mobile sink for improving lifetime in sensor networks. In *Proc. of DCOSS06*, pages 480–497, jun. 2006.

[33] Jun Luo and J.-P. Hubaux. Joint mobility and routing for lifetime elongation in wireless sensor networks. *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 3:1735 – 1746 vol. 3, mar. 2005.

[34] A. Manjeshwar, D.P. Agarwal, and H. Balakrishnan. Teen: a routing protocol for enhanced efficiency in wireless sensor networks. In *1st International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, pages 2009–2015, apr. 2001.

[35] Mirela Marta and Mihaela Cardei. Improved sensor network lifetime with multiple mobile sinks. *Pervasive and Mobile Computing*, 5(5):542 – 555, 2009.

[36] Anthony J. Nicholson and Brian D. Noble. Breadcrumbs: forecasting mobile connectivity. In *Proc. of the 14th ACM Intl. Conf. on Mobile computing and networking, MobiCom'08*, pages 46–57, New York, NY, USA, 2008. ACM.

[37] C. Park and K. W. Lee. A route maintaining algorithm using neighbor table for mobile sinks. *Wireless Networks*, 15:541–551, may. 2009.

[38] Richard W.N. Pazzi and Azzedine Boukerche. Mobile data collector strategy for delay-sensitive applications over wireless sensor networks. *Computer Communications*, 31(5):1028 – 1039, 2008.

[39] R.W. Pazzi, Dipu Zhang, A. Boukerche, and L. Mokdad. E-trail: Energy-efficient trail-based data dissemination protocol for wireless sensor networks with mobile sinks. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–5, jun. 2011.

[40] G.J. Pottie and W.J. Kaiser. Embedding the internet: wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, may. 2000.

[41] C. Schurgers and M.B. Srivastava. Energy efficient routing in wireless sensor networks. In *Military Communications Conference, 2001. MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force. IEEE*, volume 1, pages 357–361, 2001.

[42] J. H. Shin, J. Kim, K. Park, and D. Park. Railroad: virtual infrastructure for data dissemination in wireless sensor networks. In *Proceedings of the 2nd ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN05)*, pages 168–174, 2005.

[43] M. Stemm and R.H Katz. Measuring and reducing energy consumption of network interfaces in handheld devices. In *IEICE Transaction and communication*, pages 1125–1131, aug. 1997.

[44] D. Vass and A. Vidacs. Positioning mobile base station to prolong wireless sensor network lifetime. In *Proc. of ACM CoNEXT05*, pages 300–301, oct. 2005.

[45] Z. Vincze, A. Vidacs, D. Vass, R. Vida, and A. Telcs. Adaptive sink mobility in event-driven multi-hop wireless sensor networks. In *Proceedings of the 1st International Conference on Integrated Internet Ad hoc and Sensor Networks*, may. 2006.

[46] G. Wang and T. Wang. Local update-based routing protocol in wireless sensor networks with mobile sinks. In *Proceedings of the 2007 IEEE international conference on communications*, pages 3094–3099, Glasgow, Scotland, UK, 2007.

[47] Guojun. Wang, Tian Wang, Weijia Jia, Minyi Guo, and Jie Li. Adaptive location updates for mobile sinks in wireless sensor networks. *The Journal of Supercomputing*, 47(2):127 –145, feb. 2009.

[48] Xiaobing Wu and Guihai Chen. Dual-sink: Using mobile and static sinks for lifetime improvement in wireless sensor networks. In *Computer Communications and Net-*

works, 2007. ICCCN 2007. Proceedings of 16th International Conference on, pages 1297 –1302, aug. 2007.

[49] Y. Xu, J. Hendemann, and D. Estrin. Adaptive energy-conserving routing for multihop ad hoc networks. In *Technical Report TR-2000-527*, oct. 2000.

[50] H. L. Xuan and S. Lee. A coordination-based data dissemination protocol for wireless sensor networks. In *Proc. of Intelligent Sensors, Sensor Networks and Information Processing Conference*, pages 13–18, 2004.