NOTE TO USERS

This reproduction is the best copy available.

uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Wei Hong Ma

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical and Computer Engineering)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Performance Evaluation and Comparison of Routing Protocols in Wireless Sensor Network with Mobile Sinks

TITRE DE LA THÈSE / TITLE OF THESIS

A. Boukerche

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

Richard Lo

Abdulmotaleb El Saddik

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Performance Evaluation and Comparison of Routing Protocols in Wireless Sensor Network with Mobile Sinks

by

Wei Hong Ma

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

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

Your file Votre référence
ISBN: 978-0-494-73851-1
Our file Notre référence
ISBN: 978-0-494-73851-1

Abstract

Wireless Sensor Networks (WSNs) usually consist of a large number of sensors which have the ability to detect events, generate reports and transmit them to the sink which collects the data reports. The mobile sink has been adopted to improve network performance of WSNs. It is necessary to inform sensors of a topological or location change of a moving sink, however frequent location updates from mobile sinks can lead to high energy consumption of sensors, and therefore shorten the network lifetime. Many protocols have been proposed in the past few years trying to achieve the energy efficiency goal in WSNs with mobile sinks. This thesis investigates six protocols: TTDD, Locator, LURP, Efficient Routing, TwinRoute and Dual-Sink. First, they are implemented in ns-2, then extensive performance evaluations are conducted. Furthermore, the experimental results are compared and analyzed. This thesis reveals both the advantages and performance issues of these protocols, and it provides the basis for future research.

Acknowledgements

It is an honor for me to study in the School of Information Technology and Engineering, University of Ottawa, especially under the supervision of Professor Azzedine Boukerche. I would also like to thank Dr. Richard Pazzi Werner, without his help I could not complete my thesis.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Advances in technologies such as micro-electro-mechanical systems, wireless communications, and embedded computing have enabled the development of small yet multifunctional sensors, which typically have sensing, processing, transceiver and power units [1]. A large-scale wireless sensor network (WSN) consists of hundreds or even thousands of such sensors which are distributed over a large area [2]. These sensors have the ability to sense environment, generate data and send reports to the sinks which collect data reports from sensors. WSNs can be used in a wide variety of applications such as habitat monitoring [3], health care [4] and smart homes [5]. One of the recent experimental wireless sensor networks was deployed in Wytham Field, Oxford, UK. It was used by zoologists in the Department of Zoology, University of Oxford to carry out wildlife conservation research work. Wireless sensors were positioned in selected places, and zoologists roamed in the area carrying PDA-type devices which acted as mobile sinks collecting data from sensors [6, 40].

1.1 Challenging Issues

Although WSNs have numerous applications, designing a good data routing protocol for wireless sensor networks is a very challenging task. A good data routing protocol has to

take into consideration many factors such as node deployment, data reporting models, network dynamics and system requirements, which are highly application dependent [7]. However, the fundamental challenge comes from the inherent nature of wireless sensors. Because wireless sensors typically run on battery power [8] and are distributed across a large geographic area, frequent battery replacement is not an option. In some cases, wireless sensors may never get their battery replaced. Not only do sensors have a limited power supply, but they also have a limited wireless transmission range [9]. They can only communicate with their neighbors in their radio range, so sensed data is usually be relayed hop-by-hop from sensors to the destination sink.

Traditional wireless sensor networks have a static network topology. The geographic positions of sensors and sinks are fixed, thus a large volume of traffic takes place in the neighborhood of static sinks when sensed data is forwarded to the sinks. Sensors surrounding static sinks use up their energies more easily than sensors that are farther away. If these sensors failed due to lack of energy, sensed data from other sensors cann't be delivered to sinks. Non-uniform energy consumption among sensors is a problem in traditional wireless sensor networks affecting the success of sinks to collect data over a long period of time. Many energy efficient data routing protocols have been proposed to improve network lifetime of WSNs with static sinks [10–22]. However, improvements are limited [23].

With the advance of technology, a mobile sink was introduced to improve the performance of wireless sensor networks such as network connectivity, coverage, and lifetime. Many protocols have been proposed to support sink mobility in the last few years [24–34]. In wireless sensor networks with mobile sinks, sinks move randomly within the geographic area or along fixed paths to collect data from sensors. When a sink moves, the set of sensors near a sink are not fixed. As a result, sensors near a sink are more randomly distributed. This resolves the problem of non-uniform energy consumption in traditional

wireless sensor networks.

With the introduction of mobile sinks, a new challenge arises. Unlike the static sink which is located at a fixed position in the sensor field, the mobile sink collects data at random positions depending on its movement in the sensor field. For sensors to forward data successfully to a moving sink, sensors must know a sink's current position or the correct route to the sink. The position information or the route needs to be accurate, otherwise sensors may forward data to a wrong or out-of-date sink position resulting in the loss of data.

The basic idea is to let a mobile sink broadcast a beacon periodically to signal its presence. The beacon contains the sink's position information and other information such as the beacon expiry time and the beacon sequence number. After sensors near the sink receive the beacon, they will communicate with other sensors to set up the forwarding route. In order for sensors to have correct routes to a moving sink, the routing information needs to be propagated in the sensor field frequently. Inevitably, frequent routing updates consume a lot of the sensors' energy. Due to the limited energy of sensors, the energy consumption of the sensors needs to be very efficient to prolong network lifetime, which requires a well designed data routing protocol.

1.2 Contribution

This thesis provides a in-depth study of six data routing protocols which have been proposed in recent years. They are TTDD [35], Locator [36], LURP [37], Efficient Routing [38], Dual-Sink [39] and TwinRoute [40]. The contributions of this thesis are as follows.

- Five protocols were implemented using the simulation tool ns-2 [41]. The exception was TTDD due to its available source codes. Because the performance of these

protocols were evaluated by different simulation tools in the original papers. For example TTDD was simulated by ns-2, TwinRoute was simulated in TOSSIM [42], Efficient Routing was simulated in MATLAB [43].

- An extensive set of performance evaluation experiments were conducted for the six protocols, and the experimental results are presented. Moreover, a comparison analysis is provided, which also points out the advantages and performance issues of these protocols.

1.3 Thesis Organization

The rest of the thesis is organized as follows.

- Chapter 2 introduces each protocol and provides the detailed algorithm description.
- Chapter 3 presents the results of the performance evaluation of each protocol.
- Chapter 4 provides a deep comparison analysis of the performance of these six protocols.
- Chapter 5 concludes this thesis, followed by some suggestions for future research.

Chapter 2

Routing protocols of WSN with mobile sinks

In this chapter, six routing protocols of WSNs with mobile sinks are presented. The earliest protocol, TTDD was proposed in 2005, followed by Locator in 2006. LURP, Efficient Routing and Dual-Sink were proposed in 2007. While TwinRoute, the newest protocol was proposed in 2009. Unlike other protocols such as [24–28], these six protocols do not assume any fixed or controlled sink movement, instead mobile sinks move on arbitrary paths. They can be divided into two categories: location-based and topology-based protocols. TTDD, Locator and LURP are location-based protocols. The requirement of these protocols is that both sensors and sinks need to know their own geographic locations, so their basic communication mechanism is based on simple greedy geographic forwarding [44]. The location information can be acquired using a GPS receiver or through techniques like [45–47]. Efficient Routing, TwinRoute and Dual-Sinks are all topology-based routing protocols, and the sensors and sinks do not need to know their geographic locations in the network. The following sections introduce each protocol, and give a detailed explanation of its algorithm.

2.1 TTDD

2.1.1 Overview

The authors of TTDD [35] proposed a unique two-tier data dissemination approach for sinks to collect data from sensors. Instead of waiting for location updates or routing updates from mobile sinks, which is the method used by the other five protocols, a source node proactively builds a grid structure before data delivery. A grid comprises of data dissemination nodes which are sensors whose locations are nearest to the grid points. A Request for data from a sink goes through two tiers to arrive at a source node. Then, data from the source goes through the same two tiers but in the reverse order to reach the sink. The higher tier is made of the data dissemination nodes on the grid. The lower tier is within a local area of a sink's current location.

2.1.2 Algorithm Description

Algorithm for Data Announcement

As soon as a source node generates data, it builds a grid structure to prepare for data delivery. A sensor field can be divided into a grid of cells. Each cell has a size of AxA square unit. A source node is at one crossing point of the grid. It can calculate the locations of its surrounding crossing points of the grid given the knowledge of its own location (x,y) and the cell size (A). These points are called the DP (dissemination point). The source node sends out a data announcement packet to each of its adjacent DPs using simple greedy geographic forwarding. The data announcement packet has three important attributes: src_loc, which represents the location of the source node; dst_loc, which represents the destination DP's location; and seq_no, which represents the sequence number. The message will stop at the sensor node which is closer to the dissemination point than any of its neighbors. This sensor node becomes the DN (dissemination node) serving a specific DP. The DN then forwards the data announcement packet to

set up other DNs. After this recursive propagation of the data announcement packet in the sensor field, each DP on the grid is served by a DN. Each DN stores the location of its upstream DN from which it receives the data announcement message. Duplicated packets can be identified by the sequence number in the packet head and can simply be discarded. Algorithm 1 shows the main steps executed when a sensor node receives a data announcement packet. Figure 2.1 shows the grid of a source node. The gray nodes around the crossing points of the grid are the data dissemination nodes.

Algorithm 1 sensor nodes handle data announcement packet

▷ **Variables:**

upstream_dn_loc {the location of upstream DN}
data_announ_pkt {includes attributes src_loc, dst_loc, seq_no}

Action:

if data_announ_pkt seqno is duplicate or old **then**
    return,
end if
if has a neighbor sensor closest to data_announ_pkt dst_loc **then**
    forward data announcement packet to this neighbor sensor,
    return,
else
    this node becomes DN, send DN DECLARE packet to all its neighbor,
    {store the location of upstream DN}
    upstream_dn_loc = data_announ_pkt src_loc,
    send data announcement packet to its neighbor DPs,
end if

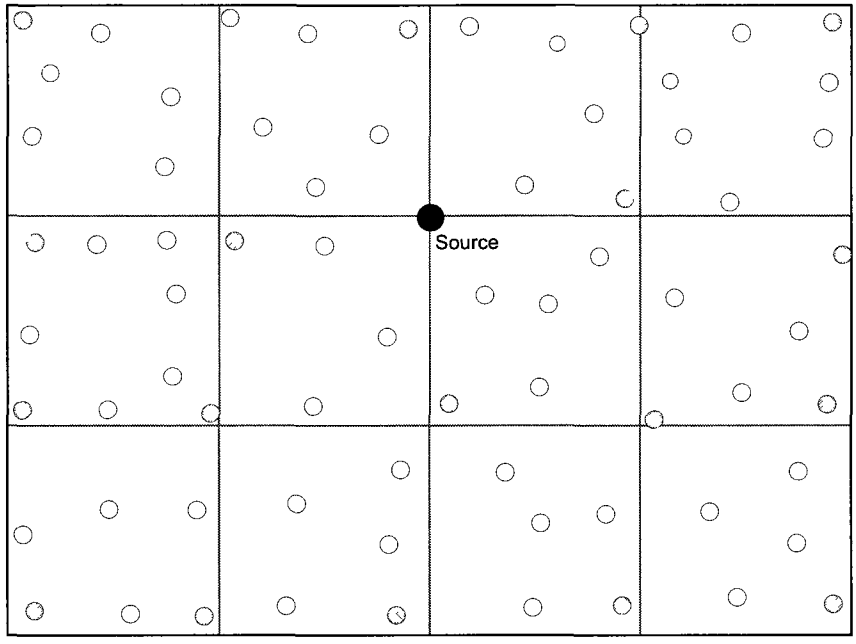


Figure 2.1: Source node and data dissemination nodes

Algorithm for Upstream Update

When a sink needs data, it sends out a query packet. The packet is flooded within a local area about the size of a cell to look for a local DN. When a DN receives the query packet, it will send an upstream update packet to its upstream DN. The upstream update packet has two important attributes: up_dn_loc, which represents the location of the upstream DN; and down_dn_loc, which represents the location of the downstream DN. The downstream DN is also the sender of the upstream packet. The upstream update packet will be forwarded through a series of data dissemination nodes on the grid until it finally reaches the source. During the forwarding process, each DN stores the location of the downstream data dissemination node from which it receives the upstream update packet. The upstream update packet is sent periodically by data dissemination nodes, and it will not be sent when a sink no longer needs data. If a source does not receive an upstream update for a while, it will stop sending data. If a sink moves more than a cell away from its previous location, the above sink query flooding and upstream update

forwarding process will be repeated. The source will send data to the sink again when it receives the newer upstream update packet.

Algorithm 2 shows the main steps executed when a sensor node receives an upstream update packet. Figure 2.2 shows the lower tier operation of a sink flooding query in a local area. DN A receives the query packet and forwards sink's query in the upstream update packet to its upstream DN B. After receiving the packet, DN B continues to forward the packet to the source. Once the source receives the upstream update packet from DN B, it will send out its data to DN B. After receiving the data packet, DN B continues to forward it to DN A. Finally, DN A will finally relay data to the sink.

Algorithm 2 sensor nodes handle upstream update packet

▷ **Variables:**

1: downstream_dn_loc {the location of downstream DN}
2: upstream_dn_loc {the location of upstream DN}
3: data_announ_pkt {includes attributes down_dn_loc, up_dn_loc}

Action:

4: **if** upstream_update_pkt up_dn_loc != myself_loc **then**
5: forward the packet to a neighbor nearest to upstream_update_pkt up_dn_loc,
6: **else**
{store the location of downstream DN}
7: downstream_dn_loc = upstream_update_pkt down_dn_loc,
{modify the attributes of the received upstream_update_pkt}
8: upstream_update_pkt down_dn_loc = myself_loc,
9: upstream_update_pkt up_dn_loc = upstream_dn_loc,
{after modification, send it}
10: send_upstream_update(),
11: **end if**

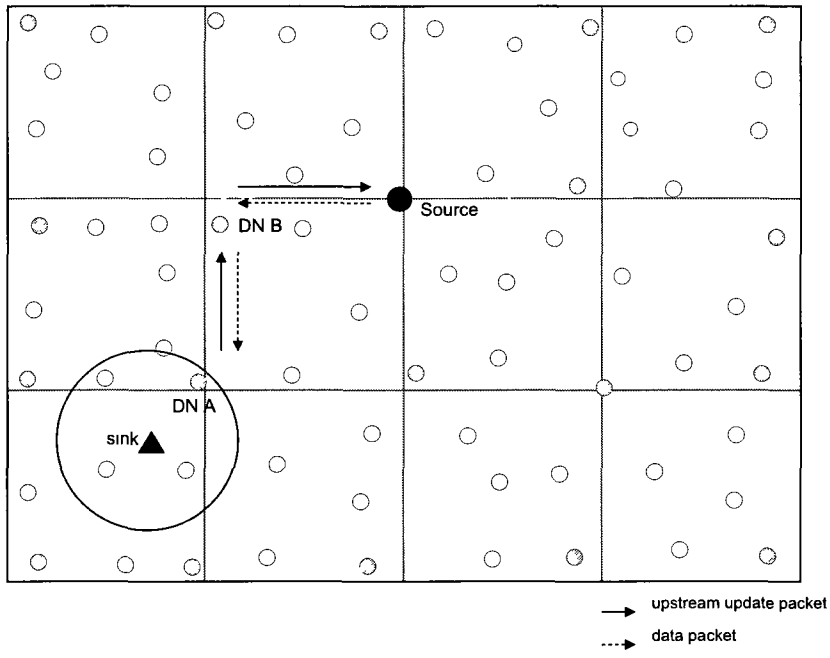


Figure 2.2: Upstream update and data delivery

2.2 Locator

2.2.1 Overview

The authors of Locator [36] proposed locators to support sink mobility. Locators are normal sensors, but are informed of the sink's current location and are responsible to answer location queries from other sensors. They are uniformly distributed in the sensor field by using a geographic hash function [48] and structured replication [48]. Using acquired sink position information from locators, a source node can deliver its sensed data to the sink.

2.2.2 Algorithm Description

Algorithms for Location Update

A sink broadcasts the location update packet, which has five important attributes: sink_id, which represents the identifier of the sink who originates the location update packet; sink_pos, which represents the current sink position; expiry_time, which represents the expiry time of current sink position; loc_pos, which represents the the position of locator calculated by geographic hashing function; and loc_forward_hops, which represents the the maximum forwarding hops for Location Update packet. Each sensor builds and maintains one routing entry for each mobile sink. Each mobile sink entry contains four important fields: sink_id, sink_pos, expiry_time and loc_pos, which are corresponding to the first four attributes in the location update packet.

As shown in Algorithm 3, a sink checks its location periodically, If its location has changed over some minimal distance or the previous location update is expired, it will send a location update packet to its immediate locators. As shown in Algorithm 4, when a sensor node receives a location update packet, it compares its own position with the locator position in the received packet, as well as its neighbor's position with the locator position. If no other sensor nodes are closer to the locator position, than it becomes the locator. If it is not the closest sensor node, it will continue to forward the packet to one of its neighbors whose position is nearest to the locator position in the received packet. If it is a locator, it updates its sink entry, then continues to send out a location update packet to other locators if the forwarding hop count is greater than 0. Figure 2.3 shows that a sink sends out its location update packet to its four surrounding locators, and the other locators are informed by these immediate locators.

Algorithm 3 Sink broadcasts location update packet

▷ **Variables:**

1: loc_update_pkt {includes attributes sink_id, sink_pos,expire time,loc_pos, loc_forward_hops}

Action:

2: **if** sink's position change >MINIMAL_CHANGED_DISTANCE **or** NOW >location_expire_time **then**
{prepare the loc_update_pkt}
3: loc_update_pkt expire_time = NOW + LOC_EXPIRE_PERIOD,
4: loc_update_pkt sink_id = myself_id,
5: loc_update_pkt sink_pos = current position,
6: loc_update_pkt loc_forward_hops = MAX_Forward_Hops,
7: get position of neighbor locators by geographic hash function,
8: **for** each neighbor locator **do**
9: loc_update_pkt loc_pos = calculated locator position,
10: send_loc_update_pkt(),
11: **end for**
12: **end if**

Algorithm 4 Sensor nodes receive location update packet

▷ **Variables:**

1: loc_update_pkt {includes attributes sink_id, sink_pos,expire time,loc_pos, loc_forward_hops}
2: sink_entry {includes fields sink_id, sink_pos,expire time,loc_pos}

Action:

3: **if** has a neighbor node closer to loc_update_pkt loc_pos than myself **then**
4: forward location update packet to this neighbor,
5: **else**
6: This sensor becomes locator,
7: update_sink_entry(),
8: ph loc_forward_hops−,
9: **if** ph loc_forward_hops >0 **then**
10: get position of neighbor locators by geographic hash function,
11: **for** each neighbor locator **do**
12: loc_update_pkt loc_pos = calculated locator position,
13: send_loc_update_pkt(),
14: **end for**
15: **end if**
16: **end if**

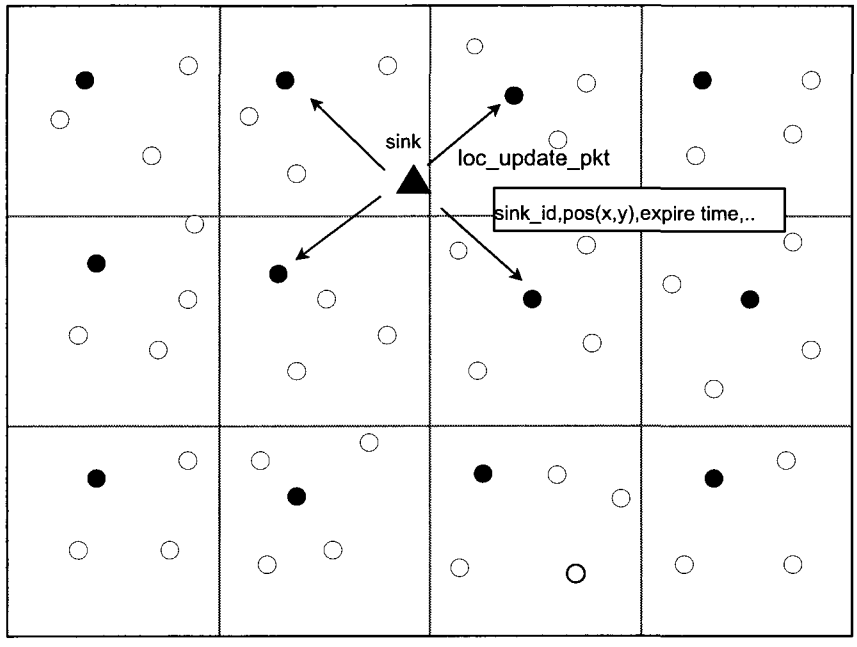


Figure 2.3: Sink broadcasts location update packet

Algorithms for Location Query and Reply

As shown in Algorithm 5, when a sensor node has data to report to a sink, and it does not know the sink location or the last location information has expired, it sends the location query packet to its nearby locators. As shown in Algorithm 6, when a sensor node receives the location query packet, if it is a locator, it sends the location reply packet back to the query sensor if it has the current sink information. Figure 2.4 shows the exchange of location query and location reply packets between a source node and a locator.

Algorithm 5 Sensor nodes send location query packet

▷ **Variables:**

1: loc_query_pkt {includes attributes sink_id, query_sensor_id, query_sensor_pos, loc_pos}

Action:

2: **if** last sink location expired **or** has no sink entry **then**
{prepare the loc_query_pkt}
3: set loc_query_pkt sink_id to the id of sink to query,
4: loc_query_pkt query_sensor_id = myself_id,
5: loc_query_pkt query_sensor_pos = myself_loc,
6: get position of neighbor locators by geographic hash function,
7: **for** each neighbor locator **do**
8: loc_query_pkt loc_pos = calculated locator position,
9: send_loc_query_pkt(),
10: **end for**
11: **end if**

Algorithm 6 Sensor nodes receive location query packet

▷ **Variables:**

1: loc_query_pkt {includes attributes sink_id, query_sensor_id, query_sensor_pos, loc_pos}
2: loc_reply_pkt {includes attributes sink_id, sink_pos,expire time,loc_pos, }
3: sink_entry {includes fields sink_id, sink_pos,expire time,loc_pos}

Action:

4: **if** I am locator **then**
5: **if** I have current sink postion for loc_query_pkt sink_id **then**
{prepare the loc_reply_pkt}
6: loc_reply_pkt sink_id = sink_entry sink_id,
7: loc_reply_pkt sink_pos = sink_entry sink_pos,
8: loc_reply_pkt expire time = sink_entry expire time,
9: loc_reply_pkt loc_pos = myself_loc,
10: loc_reply_pkt query_sensor_id = loc_query_pkt query_sensor_id,
11: loc_reply_pkt query_sensor_pos = loc_query_pkt query_sensor_pos,
12: send_loc_reply_pkt(),
13: **else**
14: **if** has a neighbor closest to loc_update_pkt loc_pos **then**
15: forward location query packet to its neigbhor,
16: **end if**
17: **end if**
18: **end if**

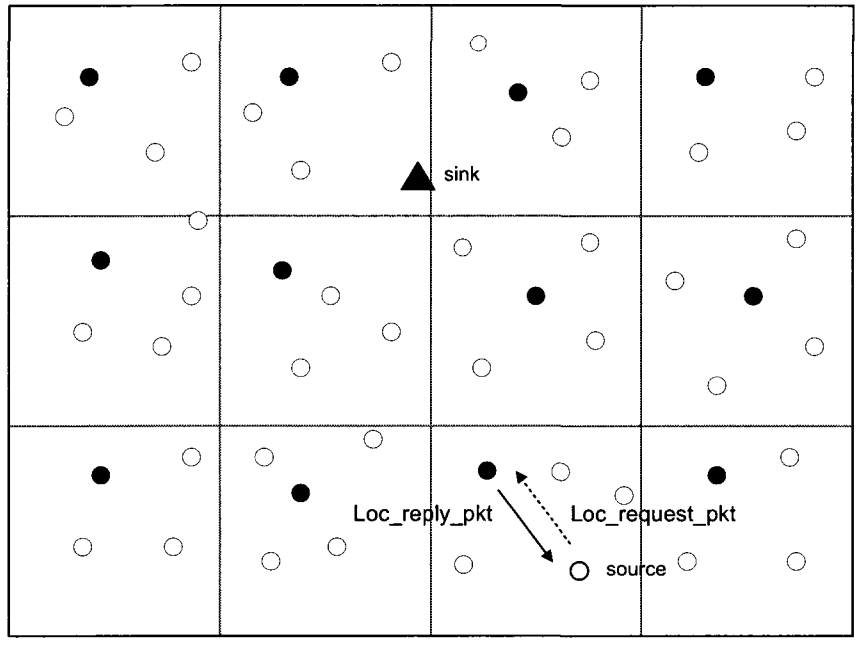


Figure 2.4: Sensor's location query and reply

Data delivery

After receiving the location reply packet, the source node will update its sink entry with the received sink's position and expiry time. Then, using the acquired sink position information, the source node can deliver its sensed data to the sink through simple greedy geographic forwarding.

2.3 LURP

2.3.1 Overview

The authors of LURP [37] suggested that when a sink moves, instead of flooding the entire network with its new location information, it only needs to broadcast its location information in a local area. The sensor field can be divided into different areas. At the start of the network operation, a sink broadcasts its location information to all the sensors, so every sensor knows in which area the sink is currently located. After that,

if it is moving inside the area, the sink will update its location information only to the sensors within the area. When a sensor needs to report data, although it does not know the accurate location of the sink, it knows the area where the sink is current visiting. Data is first delivered to the particular area, then it is forwarded to the sink within the area.

2.3.2 Algorithm Description

A sink broadcasts the location update packet, which has five important attributes: sink_id, which represents the identifier of the sink who originates the location update packet; sink_pos, which represents the current sink position; area_id, which represents the area that the sink is currently located in; update_seq, which represents the sequence number of the location update packet; and cross_area, which is a flag that can be turned on by the sink. Each sensor builds and maintains one routing entry for each mobile sink. Each mobile sink entry contains four important fields: sink_id, sink_pos, area_id, and seq_num, which correspond to the first four attributes in the location update packet.

As shown in Algorithm 7, the cross_area flag can be turned on when a sink detects that it has moved out of the previous area by checking its current position. If a sink is just moving inside of one area, it turns off the cross_area flag in its location update packet. As shown in Algorithm 8, when a sensor receives a location update packet, it checks to see if the sink entry exists. If the sink entry does not exist, it adds the sink entry, otherwise it updates the sink entry. The sensor will check its own area identifier and the received area identifier in the packet. If they are the same, it continues to broadcast the location update packet. If the cross_area flag is turned on, it will also broadcast the packet.

Algorithm 7 sink sends location update packet

▷ **Variables:**

1: loc_update_pkt {includes attributes sink_id, sink_pos, area_id, cross_area, update_seq}

Action:

2: **if** the first Location Update **then**
3: loc_update_pkt cross_area = true,
4: **else**
5: **if** in a different area **then**
6: loc_update_pkt cross_area = true,
7: **else**
8: loc_update_pkt cross_area = false,
9: **end if**
10: **end if**
{prepare for the loc_update_pkt}
11: loc_update_pkt sink_id = myself_id,
12: loc_update_pkt sink_pos = current location,
13: loc_update_pkt area_id = current area id,
14: loc_update_pkt seq_num = seq_num++,
{broadcast the packet}
15: broadcast_loc_update_pkt(),

Algorithm 8 sensor nodes receive location update packet

▷ **Variables:**

1: loc_update_pkt {includes attributes sink_id, sink_pos, area_id, cross_area, update_seq}
2: sink_entry {includes fields sink_id, sink_pos,area_id,seq_num}

Action:

3: **if** sink entry exist **then**
4: update_sink_entry(),
5: **else**
6: add_sink_entry(),
7: **end if**
8: **if** myself_area_id == loc_update_pkt area_id **or** loc_update_pkt cross_area == true **then**
9: broadcast_loc_update_pkt(),
10: **end if**

Figure 2.5 shows that in the beginning, a sink broadcasts the location update packet which is propagated throughout the sensor field due to the cross area flag being turned on by the sink. Every sensor node knows the sink's initial area identifier and position.

Figure 2.6 shows the sink moving inside the area 3, and this time it broadcasts the location update packet with its cross_area flag turned off. For example, when node A in area 3 receives the location update packet from the sink, it will continue to broadcast the location update packet because it is in the same area with the sink,. Although sensor node B in area 2 can receive the location update packet from the sink, but it is in a different area from the sink, and the cross area flag is off, so it simply drops off the packet. The result is that every sensor node in area 3 will get updated with the sink current position(x1, y1), while other sensor nodes retain the old information that the sink is in area 3 with its position(x,y).

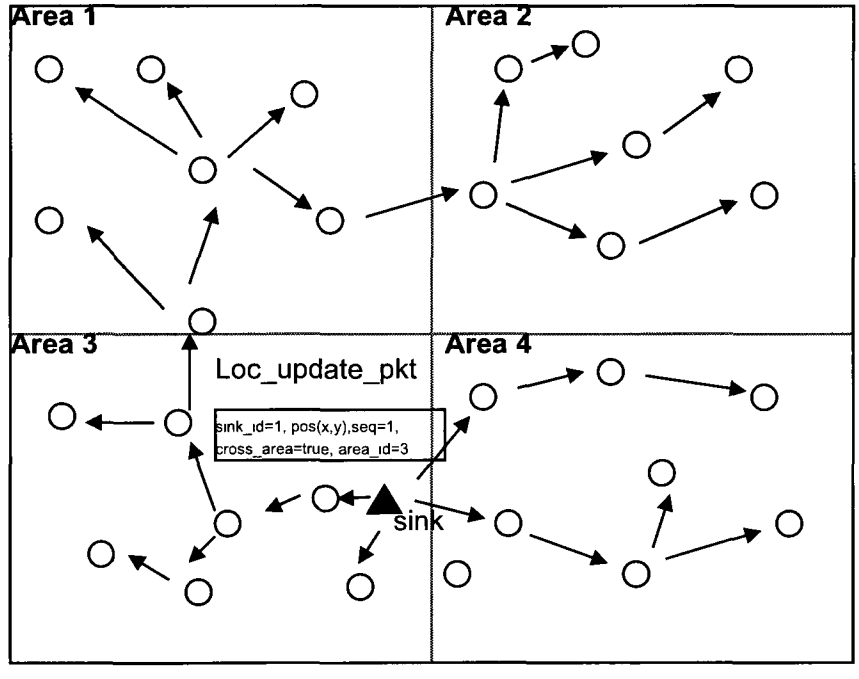


Figure 2.5: Sink broadcasts location update packet with cross area flag turned on

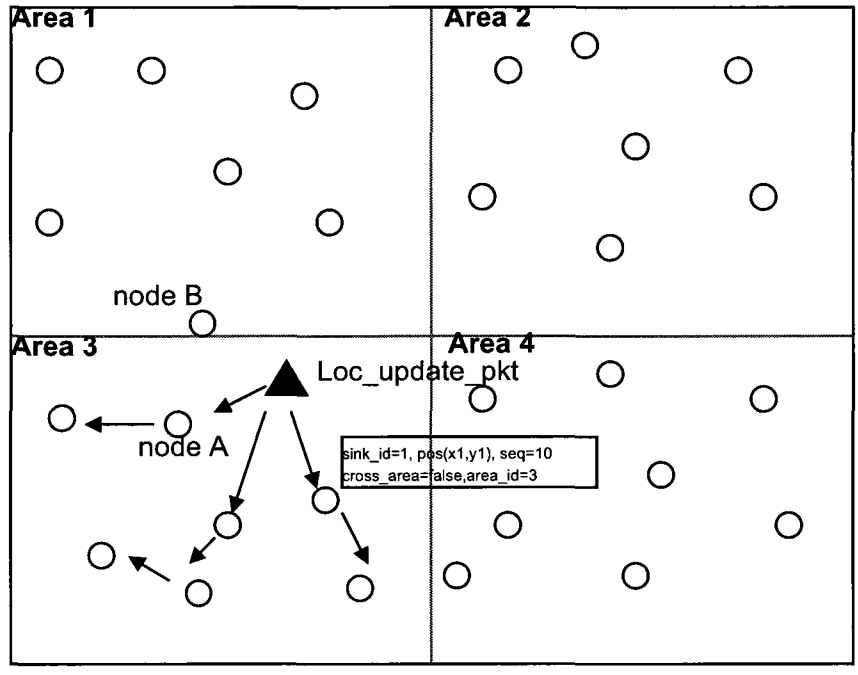


Figure 2.6: Sink broadcasts location update packet with cross area flag turned off

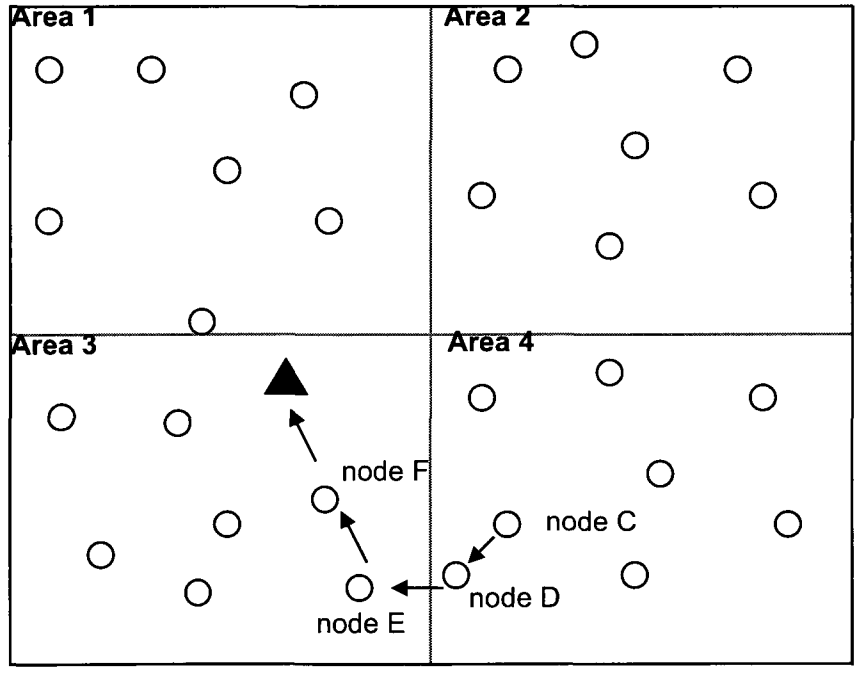


Figure 2.7: Data delivery from the source node to the sink

Data delivery

When a sensor forwards data, if it is in a different area from the sink, it will send data to the area where the sink is. For example, in Figure 2.7, when node C has data to report, it forwards its data to node D because node D is its neighbor which is closer to the sink position(x,y). Node D then forwards the data to node E, because node E is its neighbor which is closer to the sink position(x,y). Node E knows the accurate position of sink which is pos(x1,y1) because it is in the same area as the sink. Node E forwards the data to node F because node F is its neighbor and is closer to the sink position. Node F forwards data to the sink directly.

2.4 Efficient Routing

2.4.1 Overview

The authors of Efficient Routing [38] proposed a simple yet effective routing protocol which limits the propagation of the routing updates in the sensor field. A mobile sink sends out a beacon periodically to let nearby sensors know of its presence, thus a routing tree can be built with the sink as the root. Each sensor has a routing entry which keeps a record of its distance to the sink in hop counts and its next hop neighbor in the direction of the sink. A sensor will modify its routing entry and continue to broadcast the routing update only if the change of its distance to the sink exceeds a given percentage threshold.

2.4.2 Algorithm Description

A routing update packet contains four important attributes: sink_id, which represents the identifier of a mobile sink; seq_num, which represents the sequence number that can be used to prevent a message loop; cost, which represents the distance to the sink in hop counts; and src_id, which represents the packet sender's identifier. A sink sends a routing update packet every few seconds. Each time it increases the sequence number by one,

which means that the higher the sequence number, the newer the packet is. Each sensor builds and maintains one routing entry for each mobile sink. Each mobile sink entry contains four important fields: sink_id, seq_num, cost, and next_hop sensor identifier, which correspond to the four attributes in the routing update packet.

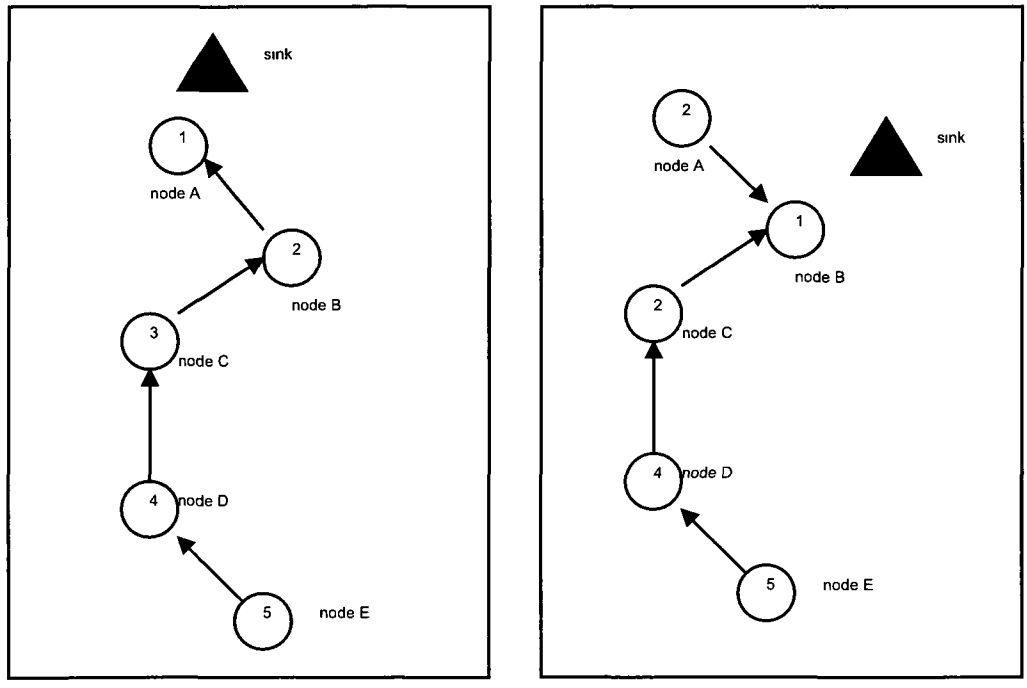
As shown in Algorithm 9, when a sensor receives the routing update packet, it first checks to see if the mobile sink entry exists. If not, the sensor adds an entry in the routing table for the sink. If the sink entry already exists, it further checks the cost change. If the cost change exceeds a given percentage threshold and the packet is a newer one, then it updates the routing entry for the sink. After the sensor node adds or modifies its routing table, it will continue to broadcast the routing update packet, with sink_id and seq_num unchanged, but increasing the cost by one and replacing the src_id using its own identifier.

Algorithm 9 Senor nodes receive the routing update packet

```
▷ Variables:
1: threshold_percent {cost change threshold in percent}
2: sink_entry {includes fields  sink_id,seq_num, cost, next_hop}
3: update_pkt {includes attributes  sink_id,seq_num, cost, src_id}

 Action:
4: if the sink_entry exists then
5:    if abs( sink_entry cost - update_pkt cost ) >( sink_entry cost * threshold_percent ) and (update_pkt seq_num
      >sink_entry seq_num) then
6:       modify_sink_entry(),
7:       update_pkt cost++;
8:       update_pkt src_id = myself_id,
9:       broadcast_update_pkt(),
10:   end if
11: else
12:   add_sink_entry(),
13:   update_pkt cost++,
14:   update_pkt src_id = myself_id,
15:   broadcast_update_pkt(),
16: end if
```

Figure 2.8 shows the change to the routing tree after a sink moves from node A to node B. Suppose the threshold percent is specified as 0.3. Before the sink moves, every sensor node points to its next hop sensor node and keeps a record of its cost to the sink. When the sink moves to node B, the routing tree gets updated and sensor nodes A, B and C update their costs to the sink. Because the previous cost for node D is four, so the cost change needs to be greater than 1.2 (4X0.3=1.2) when it receives the newer routing update from node C. If the cost changes is less than this threshold, node D will not update the sink routing entry and it will not continue to broadcast the routing packet. Node E has not yet received the newer routing packet, so it still retains the old routing information.



(a) Routing tree before sink moves (b) Routing tree after sink moves

Figure 2.8: Routing tree before and after sink moves

2.5 TwinRoute

2.5.1 Overview

The authors of TwinRoute [40] proposed a hybrid routing approach which combines two schemes. One is a proactive scheme, called P-scheme, and the other is a reactive scheme, called R-scheme. Both schemes build their own kinds of routing trees from a different network perspective. P-Scheme is designed to exploit a sink's long-term visiting pattern. It builds routing trees rooted at the storage nodes. Only those sensors that are frequently visited by sinks may become storage nodes. R-Scheme is designed to react to real-time sink movement. It builds routing trees dynamically around a moving sink. When a source node has data to report, it chooses a next hop forwarding sensor either from R-scheme or from P-scheme routing trees. Data will be forwarded to a sink along the instant R-scheme tree when a sink happens to be nearby. Alternatively, data can be forwarded to the storage node along the P-scheme tree where data will be collected by visiting mobile sink.

2.5.2 Algorithm Description

Sensors have to process two kinds of routing packets: P-scheme and R-scheme. Each kind of packet contains two important attributes: distance, which represents the number of hops to the root node; and src_id, which represents the packet sender's identifier. Each sensor node keeps four important variables: P_scheme_distance and R_scheme_distance, which correspond to the distance attribute in the P-scheme and R-scheme routing packets; and P_scheme_parent and R_scheme_parent, which correspond to the src_id attribute in the P-scheme and R-scheme routing packets.

Algorithm for P-scheme packet

The algorithm for a sensor node to process the P-scheme routing packet is relatively simple. It checks the received distance value with its stored P_scheme_distance value. If the received value is smaller, it will store the received value as its P_scheme_distance and the sender of the packet as its P_scheme_parent. After that, it will broadcast the P-scheme packet, in which the src_id is replaced with its own identifier, and the distance is increased by one.

Algorithm 10 Senor nodes receive P-scheme packet

▷ **Variables:**

1: P_scheme_distance {number of hops to the storage node}
2: P_scheme_parent {next hop node ID in the routing tree}
3: P_scheme_pkt {includes attributes distance, src_id}

Action:

4: **if** P_scheme_pkt distance <P_scheme_distance **then**
5: P_scheme_distance = P_scheme_pkt distance,
6: P_scheme_parent = P_scheme_pkt src_id,
7: P_scheme_pkt distance++,
8: P_scheme_pkt src_id = myself_id,
9: broadcast_P_scheme_pkt(),
10: **end if**

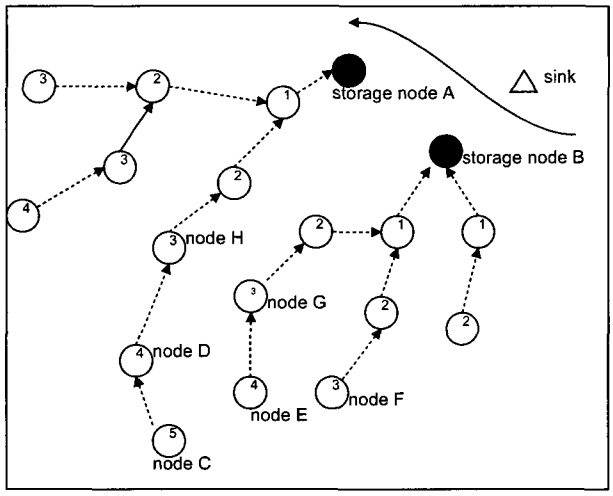


Figure 2.9: P_scheme routing trees rooted at the storage node A and B

Figure 2.9 shows the P-scheme routing trees in a sensor field. Because the sink passed sensor nodes A and B repeatedly in the past, nodes A and B become storage nodes, and two corresponding P-scheme trees are built up. Each node records its distance to the storage node in the number of hops and points to its next-hop parent sensor.

Algorithm for R-scheme packet

It is slightly more complicated to process the R-scheme packet. A sensor has two timers associated with the R-scheme, one is just_covered_timer, and the other is tree_destruct_timer. When a sensor node receives a R-scheme packet, it will check its justCovered flag to see if it has recently been part of a R_scheme tree. The sensor node also compares the received distance value with the stored P_scheme_distance value. If the received distance value is shorter than the stored P_scheme_distance by the SP (scheme preference) parameter and the flag is turned off, then the received distance is saved as R_scheme_distance and the sender of the packet is saved as its R_scheme_parent. The sensor will turn on its justCovered flag and start its two timers, so it will not participate in another R-scheme tree until its flag is turned off when the just_covered_timer expires.

When the tree_destruct_timer expires, it will destroy the current records by setting the R_scheme_distance and R_scheme_parent to unknown. If the R-scheme tree depth is still satisfied, it will broadcast the R_scheme packet, in which the src_id is replaced with its own identifier, and the distance is increased by one. If not, it will stop broadcasting the packet.

Algorithm 11 Senor nodes receive R-scheme packet

▷ **Variables:**

1: R_scheme_pkt {includes attributes distance, src_id}
2: R_scheme_distance {number of hops to the root node}
3: R_scheme_parent {next hop node ID in the routing tree}
4: P_scheme_distance {number of hops to the storage node node}
5: SP {scheme preference parameter}
6: TD {tree depth}
7: justCovered {flag indicates the recent tree coverage}

Action:

8: **if** justCovered == false **and** (P_scheme_distance - R_scheme_pkt distance) >SP **then**
9: R_scheme_distance = R_scheme_pkt distance,
10: R_scheme_parent = R_scheme_pkt src_id,
11: justCovered == true,
12: start tree_just_covered_timer,
13: start tree_destruct_timer,
14: **if** R_scheme_pkt distance + 1 <TD **then**
15: R_scheme_pkt distance++,
16: R_scheme_pkt src_id= myself_id,
17: broadcast_R_scheme_pkt(),
18: **end if**
19: **when** just_covered_timer expires:
20: resetJustCovered()
21: justCovered = false,
22: **end when**
23: **when** tree_destruct_timer expires:
24: treeDestruct()
25: R_Scheme_distance = unknown,
26: R_Scheme_parent = unknown,
27: **end when**
28: **end if**

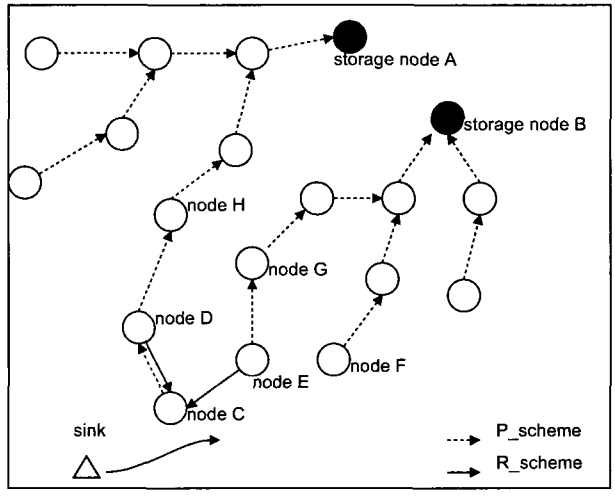


Figure 2.10: Co-exist of R_scheme and P_scheme routing trees

Figure 2.10 shows the instant R-scheme routing tree when a sink is near node C. Although the sink is not near node A or node B, the P-scheme trees are still there. In this example, the SP parameter is set to 2 and Tree Depth parameter is set to 2. When a sink happens to pass node C, node C becomes the root node of the R-scheme tree and broadcasts a R-scheme routing packet with the distance set to 1. Nodes D and E, both have the P_Scheme_distance value of 4. Because (4 - 1) is greater than 2, nodes D and E will become part of the R-scheme tree and set their timers, while they still keep the P-scheme routing information. After checking the tree depth, node D and E continue to send out the R_scheme routing packet with the distance set to 2. Nodes F, H, and G all have P_Scheme_distance value of 3. These three nodes can not satisfy the selection rule when they receive the R-scheme packet, because (3 - 2) is less than 2. Nodes F, H, and G will not become part of the R-scheme routing tree, they only keep the P-scheme routing information.

Data delivery

When a sensor forwards data, if it has both P-scheme and R-scheme routing information, it uses the R-scheme routing tree. For example, in Figure 2.10, node H will forward its data to the storage node A along the P-scheme tree. Node E can forward its data to the sink along the instant R-scheme tree. When the timer expires and the sink moves out, node E will return to using the P-scheme tree if it still has data to send.

2.6 Dual-Sink

2.6.1 Overview

The authors of Dual-Sink [39] suggested that both mobile sinks and static sinks are used in the sensor field to improve the network lifetime. At the start of the network operation, the static sink broadcasts its presence to all the sensors. A tree topology can be build up with the static sink as the root. The mobile sink needs to broadcast its presence periodically so that sensors have an updated route to the moving sink. However, the broadcast is limited to a subset of the sensors in the sensor field. The result of the deployment of both static and mobile sinks is that for every sensor there is one route leading to the static sink. At the same time, sensors will have the chance to be part of a routing tree leading towards the mobile sink according to the movement of the sink.

2.6.2 Algorithm Description

A mobile sink sends hello packets periodically. A hello packet includes four important attributes: sink_id, which represents the sink identifier; ttl, which represents the range of the broadcast; src_id, which represents the packet sender's identifier; and expiry time. Each sensor builds and maintains one routing entry for each mobile sink. Each mobile sink entry contains four important fields: sink_id, hop_num, next_hop sensor identifier, and expiry time, which correspond to the four attributes in the hello packet. The

hop_num value can be calculated from the received ttl value.

When a sensor receives a hello packet, it checks to see if the mobile sink entry exists. The sensor will update the mobile sink entry if it exists, and if not, it will add the mobile sink entry. Then the sensor will reduce the ttl value by one. If the reduced value is greater than zero, it will continue to broadcast the hello packet, replacing the src_id with its own identifier. The mobile sink entry is not valid after expiry time.

Algorithm 12 Senor nodes receive Hello packet

▷ **Variables:**

1: mobile_sink_entry {includes fields sink_id,hop_num, next_hop, expire time}
2: hello_pkt {includes attributes sink_id,ttl, src_id, expire time}

Action:

3: **if** the mobile_sink_entry exists **then**
4: update_sink_entry(),
5: **else**
6: add_sink_entry(),
7: **end if**
8: **if** (hello_pkt ttl - 1)>0 **then**
9: hello_pkt src_id = myself_id,
10: hello_pkt ttl–,
11: broadcast_hello_pkt(),
12: **end if**

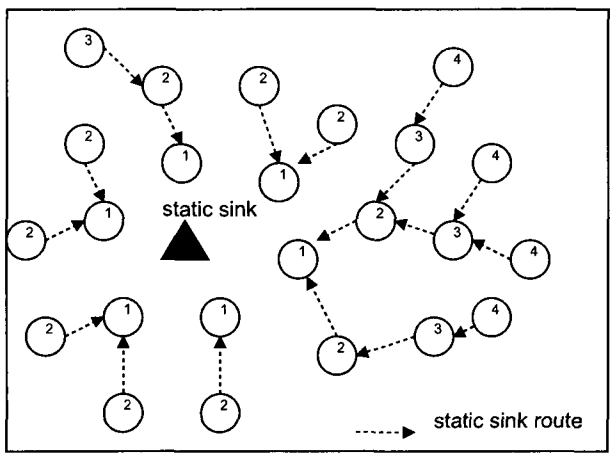


Figure 2.11: The static routing tree

As shown in Figure 2.11, a static sink broadcasts its presence in the beginning, a static routing tree is built up, every sensor node points to its next-hop sensor and records its hop distance to the sink.

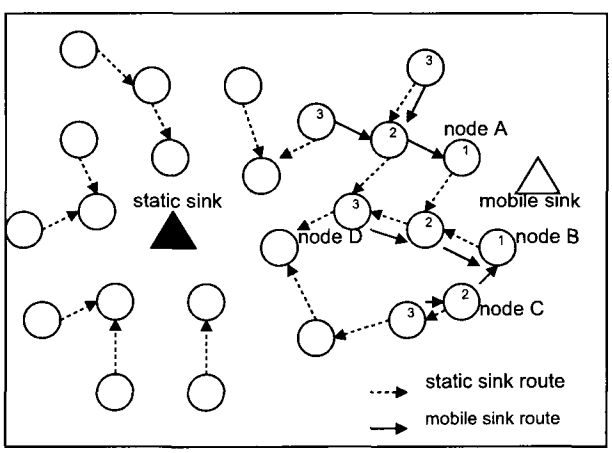


Figure 2.12: Co-exist of mobile and static routing trees

Figure 2.12 shows the mobile routing tree built around the mobile sink at the moment when the mobile sink is between node A and node B. In this example, the maximum ttl value is set to 3 by the mobile sink. Because of the limitation of the hello packet broadcast, only some sensor nodes have both static and mobile routing trees information.

Data delivery

When a sensor forwards data, if it has both the static sink routing and mobile sink routing information, it will pick up the route with less hop accounts. For example in Figure 2.12, node C is 4 hops away from the static sink, but only 2 hops away from the mobile sink, so it forwards its data to node B in the mobile routing tree. Node D is 3 hops away from the mobile sink, and 2 hops away from the static sink, so node D forwards its data to its next-hop sensor in the static routing tree. Naturally, the static route is used if a node has no route to the mobile sink.

Chapter 3

Performance Evaluation

3.1 Simulation Scenario and Metrics

The six protocols were implemented and evaluated in the ns-2 simulator. The simulation scenario consisted of 200 sensor nodes which were randomly distributed in a 2000 square meter field. Mobile sinks were randomly moving in the sensor field, and random Way-point Model [49]was the mobility pattern. The number of source nodes and the number of mobile sinks were varied to evaluate their impact on the network performance of the six protocols. The simulation parameters used in the tests are listed in Table 3.1. Note that for Dual-Sink, the number of sinks varied from 2, 4, 6, to 8, and there is only one static sink in all test configurations.

Three metrics were chosen to evaluate the performance of the six protocols. The first two are the energy consumption and the data delivery rate. A well-designed WSN routing protocol must achieve a desirable data delivery rate while making good use of sensor node's limited power supply. The third is the delay which indicates the freshness of the packet.

- Energy consumption is defined as the total energy consumption of sensor nodes spent on communications.

- Data delivery rate is defined as the ratio of the total number of successfully received data packets in all sinks to the total number of data packets generated by source nodes.
- Delay is defined as the average latency from the moment that a packet is transmitted from a source to the moment that it is received by a sink.

Table 3.1: Summary of the simulation parameters

Parameter	value
Simulation time	200s
Simulation area	2000mx2000m
Number of sensor nodes	200
Number of sinks	1,2,4,6,8
Number of source sensor nodes	4,6,8
Data rate of source sensor nodes	1pkt/s
Speed of mobile sink	10 m/s
TX power dissipation rate	0.66w
Rx power dissipation rate	0.395w
Idle power dissipation rate	0.035w
Initial energy of sensor node	1000J

In addition to the above tests, network lifetime simulation tests were performed for every protocol. The simulation parameters and the results will be presented and discussed in next chapter.

3.2 TTDD

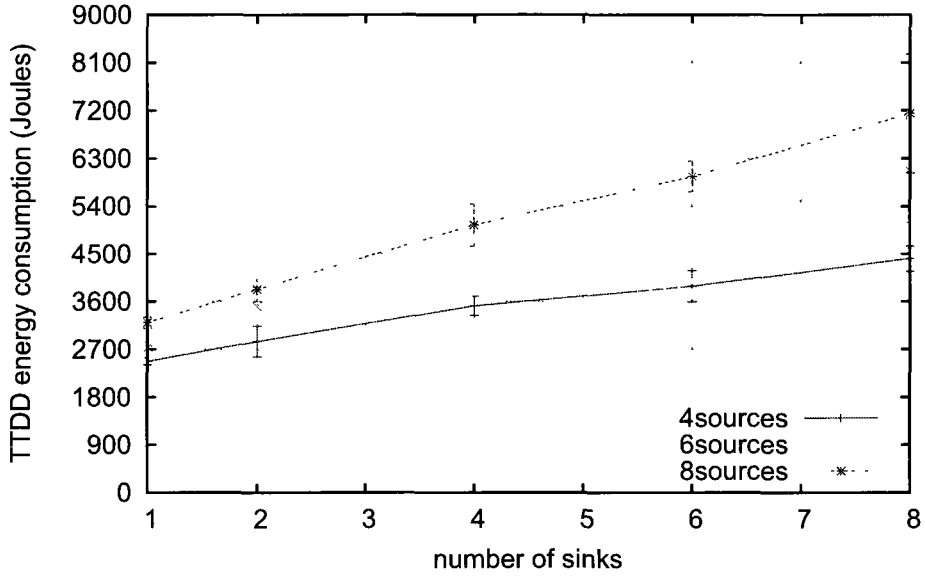


Figure 3.1: TTDD energy consumption vs. numbers of sinks and sources

Figure 3.1 shows the energy consumption of TTDD. The energy consumption goes up linearly as the number of sinks increase. When a sink needs data, it floods a local query to find a nearby dissemination node. More sinks flood more local query packets, which leads to more energy consumption.

With a fixed number of sinks, energy consumption goes up as the number of sources increase. Each source builds its own grid infrastructure, the more sources the more sensor nodes become dissemination nodes, which leads to more energy consumption.

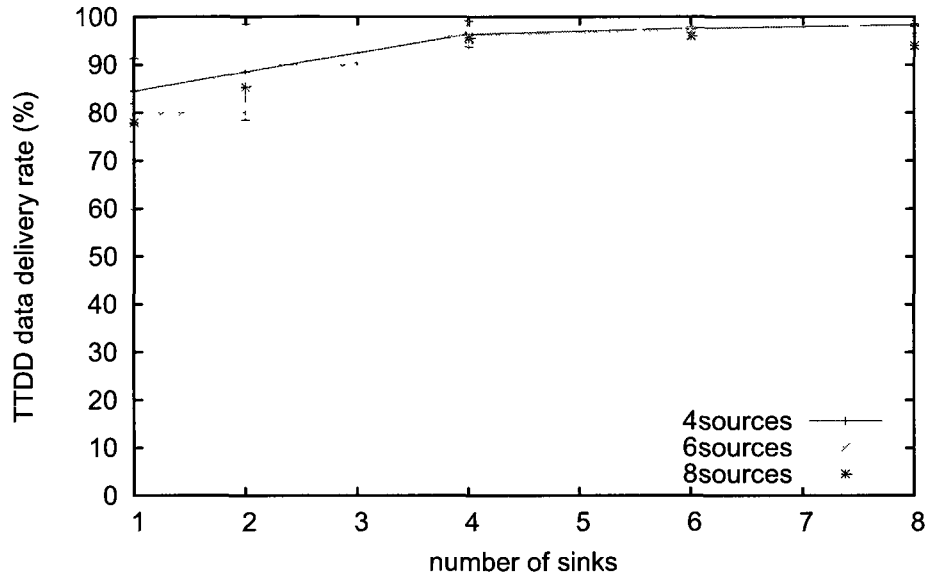


Figure 3.2: TTDD data delivery rate vs. numbers of sinks and sources

Figure 3.2 shows the data delivery ratio of TTDD. Generally speaking, the data delivery ratio goes up as the number of sinks increase. When the network is configured with 4 sinks, the data delivery ratio is above 95%. However, the data delivery ratio tends to decrease with the increasing number of sources. For example, when the network is configured with 8 sinks, the data delivery ratio goes down from 98%, 96%, to 94% as the number of sources increase from 4, 6, to 8. More sources generate more data packets, which leads to more collision and data loss [50, 51].

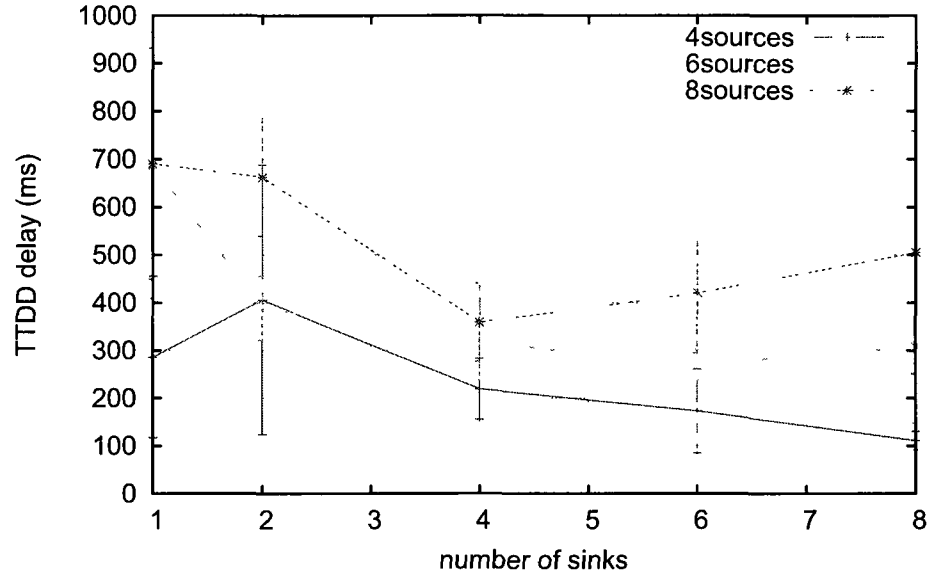


Figure 3.3: TTDD delay vs. numbers of sinks and sources

Figure 3.3 shows the delay of TTDD. Overall the delay is small, under 1 second. With a fixed number of sinks, the delay goes up as the number of sources increase. For example, when the the network is configured with 4 sinks, the delay goes up from 219ms, 318ms, to 359ms as the number of sources increases from 4, 6, to 8.

3.3 Locator

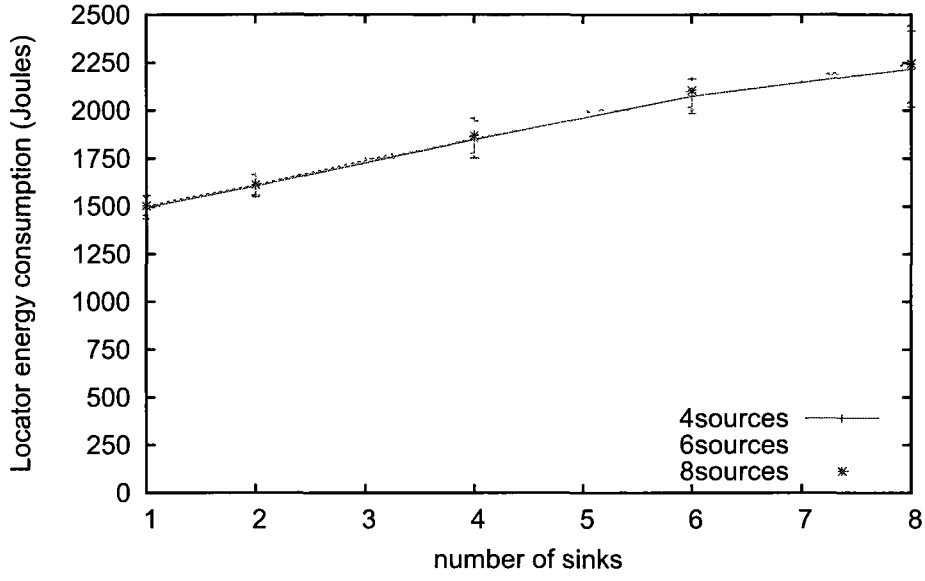


Figure 3.4: Locator energy consumption vs. numbers of sinks and sources

Figure 3.4 shows the energy consumption of Locator. The energy consumption goes up linearly as the number of sinks increase. Because more sinks send out more location update packets which are propagated in the sensor field, this leads to more energy consumption.

However the change in the number of sources has little effect on the energy consumption. For example, when the network is configured with 4 sinks, the energy consumption goes up from 1848, 1855, to 1868 as the number of sources increase from 4, 6, to 8. In Locator there is no grid-like infrastructure for each source so more sources do not introduce more energy consumption.

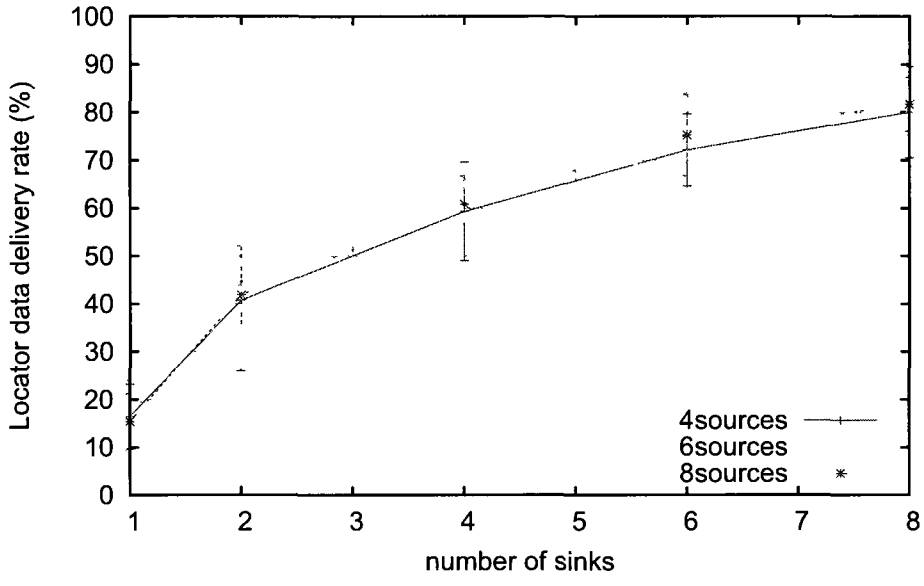


Figure 3.5: Locator data delivery rate vs. numbers of sinks and sources

Figure 3.5 shows the data delivery ratio of Locator. The data delivery ratio goes up as the number of sinks increase. When the network is configured with 1 sink, the data delivery ratio is only about 15%. As the number of sinks increase the data delivery ratio rises quickly. When the network is configured with 8 sinks, the data delivery ratio reaches 80%. The greater the number of sinks, the better the chance for data of being collected.

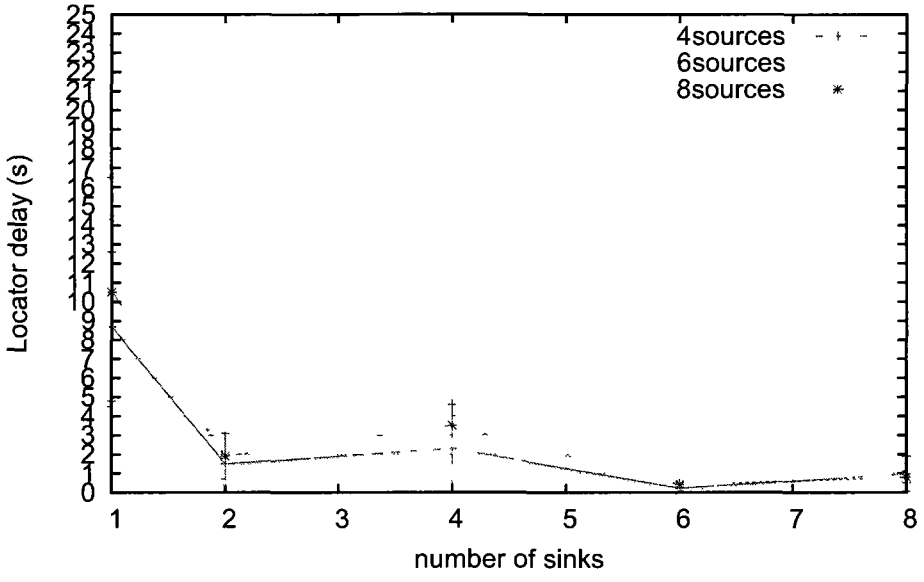


Figure 3.6: Locator delay vs. numbers of sinks and sources

Figure 3.6 shows the delay ratio of Locator. The delay is large, especially when the network is configured with a lower number of sinks. For example, when the network is configured with 1 sink and 4 sources, the delay is nearly 9 seconds. The delay decreases when the network is configured with more sinks. For instance, when the network is configured with 6 sinks and 4 sources, the delay is under 1 second.

3.4 LURP

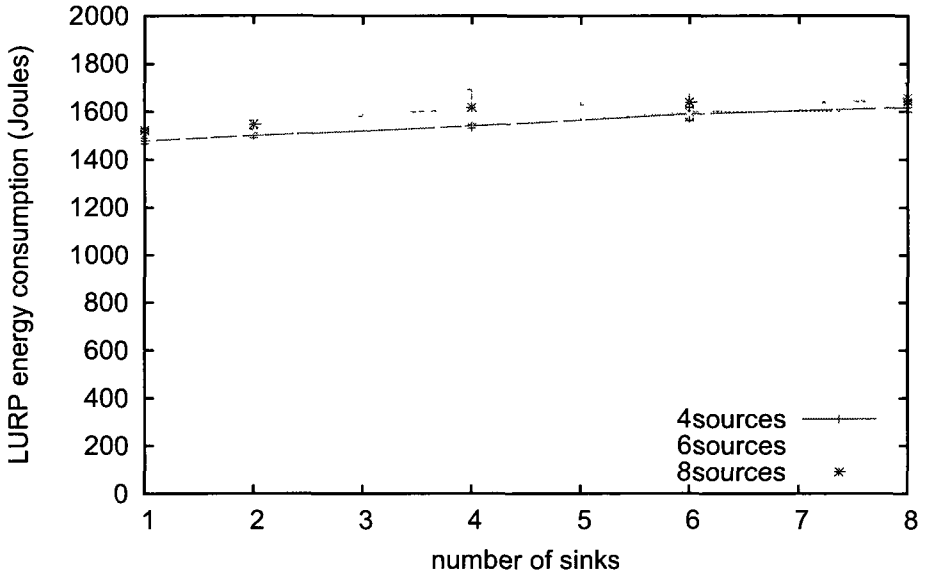


Figure 3.7: LURP energy consumption vs. numbers of sinks and sources

Figure 3.7 shows the energy consumption of LURP. The energy consumption goes up as the number of sinks increase. Because more sinks broadcast more location update packets which are propagated in the sensor field, this leads to more energy consumption.

However, the change in the number of sources has little effect on the energy consumption. For example, when the network is configured with 4 sinks, the energy consumption goes up from 1540, 1550, to 1617 as the number of sources increase from 4, 6, to 8. In LURP there is no grid-like infrastructure for each source so more sources do not introduce more energy consumption.

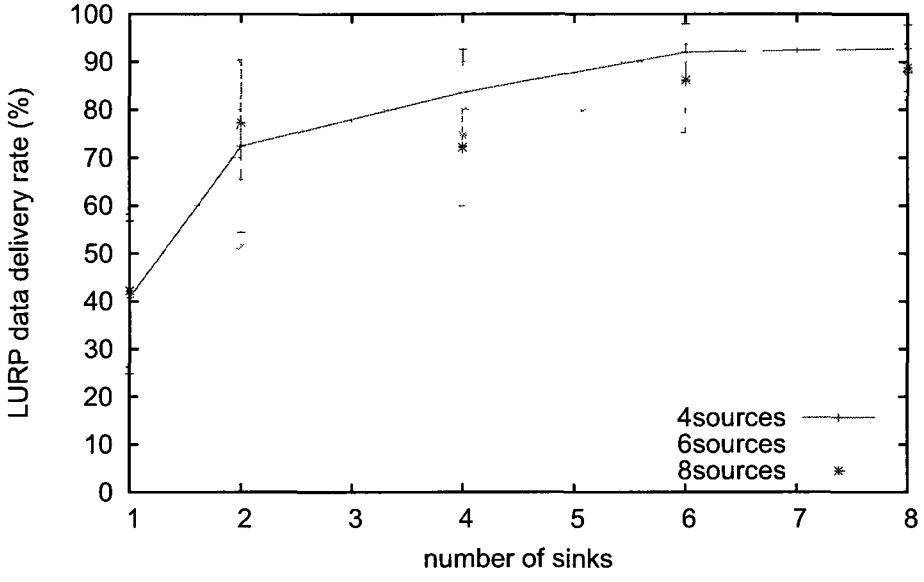


Figure 3.8: LURP data delivery rate vs. numbers of sinks and sources

Figure 3.8 shows the data delivery ratio of LURP. Generally speaking, the data delivery ratio goes up as the number of sinks increase. When the network is configured with 1 sink, the data delivery ratio is about 40%. As the number of sinks increases the data delivery ratio rises quickly. When the network is configured with 8 sinks, the data delivery ratio reaches 90%. The greater the number of sinks, the better the chance for data of being collected.

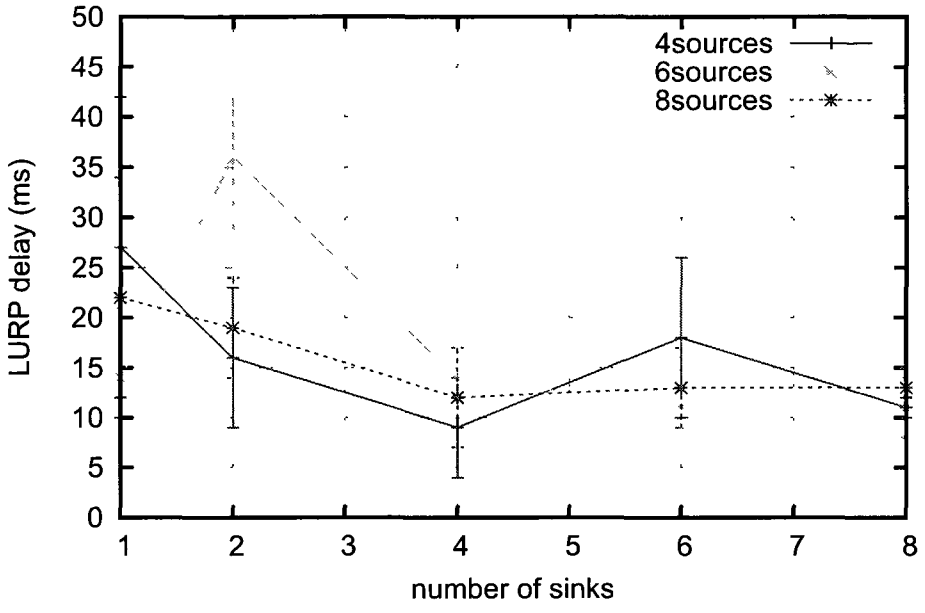


Figure 3.9: LURP delay vs. numbers of sinks and sources

Figure 3.9 shows the delay of LURP. The delay is very small, for all network configurations it is under 50ms.

3.5 Efficient Routing

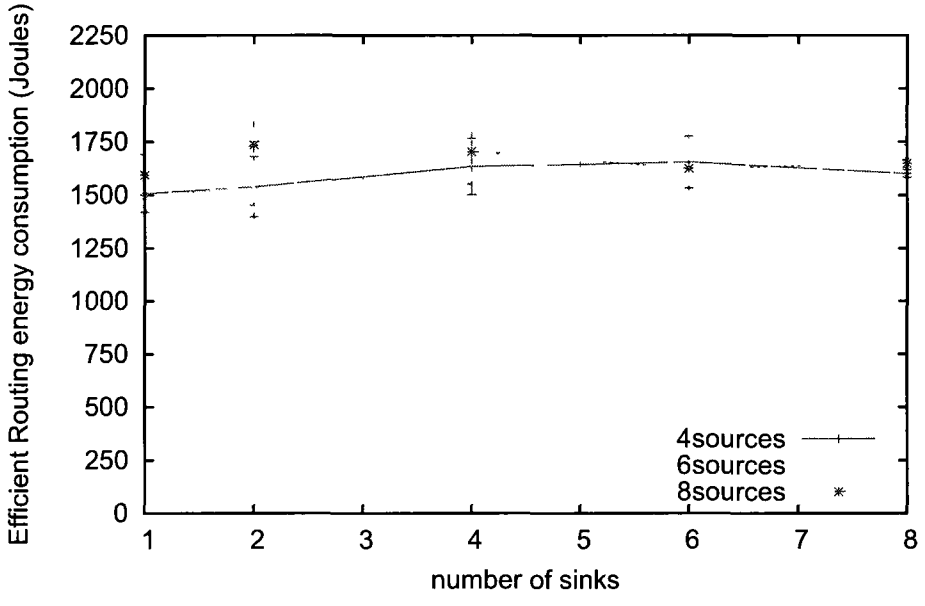


Figure 3.10: Efficient Routing energy consumption vs. numbers of sinks and sources

Figure 3.10 shows the energy consumption of Efficient Routing. The energy consumption fluctuates with changes in the number of sinks. Although each sensor builds one routing entry for every sink, the parameter threshold_percent plays a role in dynamically updating the routing entry which causes the fluctuation of the energy consumption.

The change in the number of sources has little effect on the energy consumption. For example, when the network is configured with 4 sinks, the energy consumption goes up from 1634, 1701, to 1704 as the number of sources increase from 4, 6, to 8. In Efficient Routing there is no grid-like infrastructure for each source so more sources do not introduce more energy consumption.

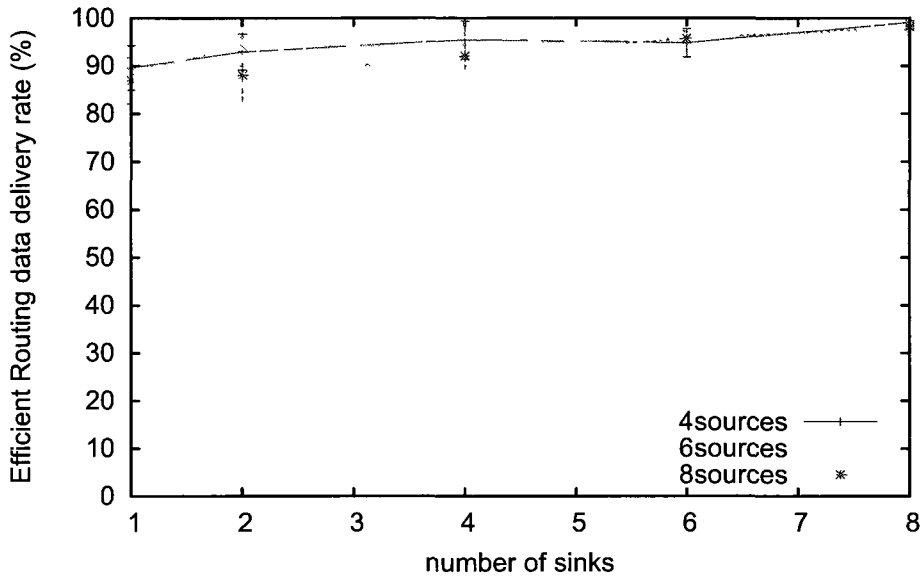


Figure 3.11: Efficient Routing data delivery rate vs. numbers of sinks and sources

Figure 3.11 shows the data delivery ratio of Efficient Routing. Efficient Routing shows very good data delivery ability. Even when the network is configured with only one sink, Efficient Routing can deliver about 88% of source data to the sinks. As the number of sinks increase, the data delivery ratio goes up. When the network is configured with 8 sinks, the data delivery ratio is above 97%. A greater number of sinks leads to a greater number of routing trees, which means that data has a better chance of being collected.

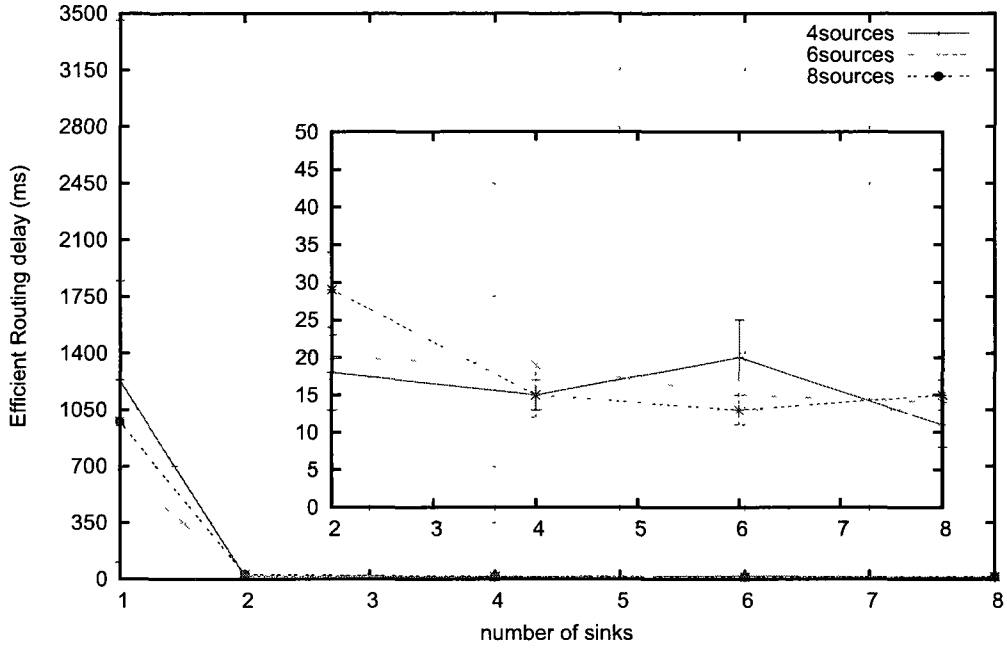


Figure 3.12: Efficient Routing delay vs. numbers of sinks and sources

Figure 3.12 shows the delay of Efficient Routing. The delay dramatically changes as the number of sinks increase. For example, when the network is configured with 1 sink and 4 sources, the delay is more than 1 second. When the network is configured with more than 1 sink, the delay is under 50ms for all configurations.

3.6 TwinRoute

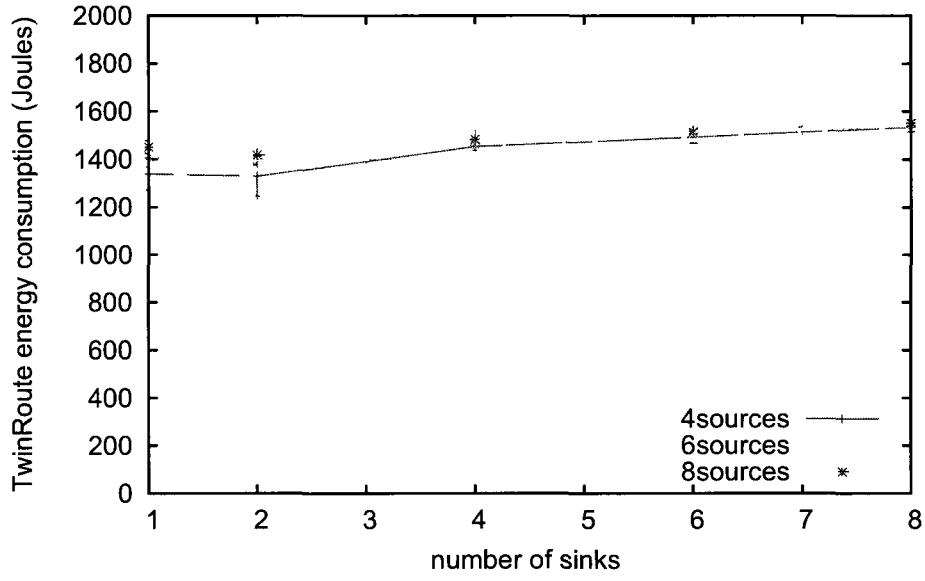


Figure 3.13: TwinRoute energy consumption vs. numbers of sinks and sources

Figure 3.13 shows the energy consumption of TwinRoute. Generally speaking, the energy consumption goes up as the number of sinks increase. A greater number of sinks leads to a greater number of routing trees, which causes more energy consumption.

However the change of the number of sources has little effect on the energy consumption. For example, when the network is configured with 4 sinks, the energy goes up from 1453, 1462, to 1484 as the number of sources increases from 4, 6, to 8. In TwinRoute there is no grid-like infrastructure for each source so more sources do not introduce more energy consumption.

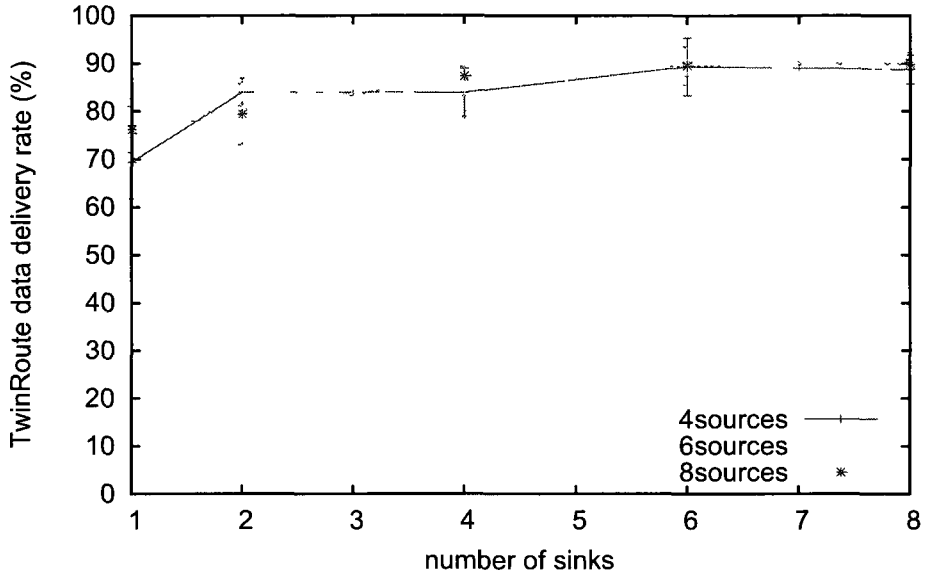


Figure 3.14: TwinRoute data delivery rate vs. numbers of sinks and sources

Figure 3.14 shows the data delivery ratio of TwinRoute. Generally speaking, the data delivery ratio goes up as the number of sinks increase. When the network is configured with 6 sinks, the data delivery ratio increases to nearly 90%. A greater number of sinks leads to a greater number of routing trees, which means that data has a better chance of being collected.

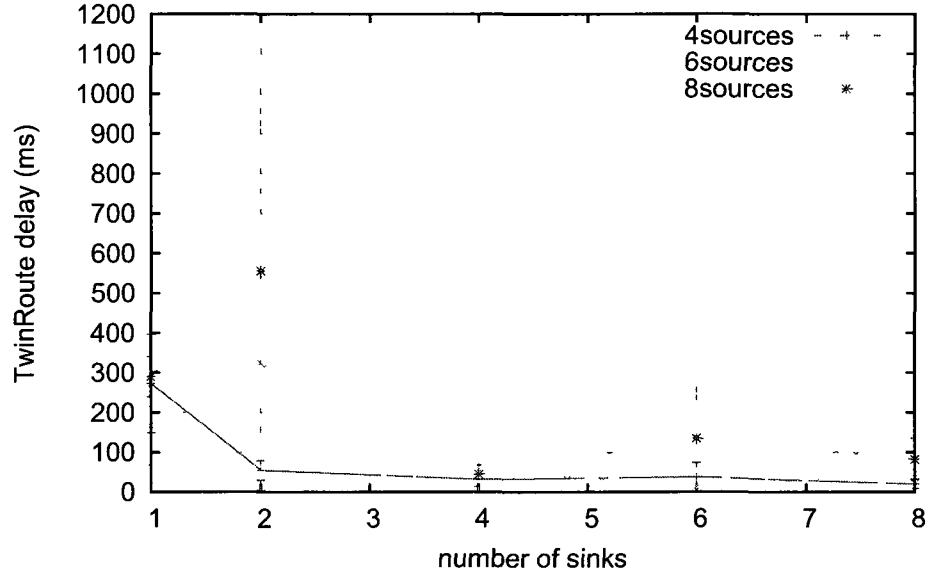


Figure 3.15: TwinRoute delay vs. numbers of sinks and sources

Figure 3.15 shows the delay of TwinRoute. Overall the delay is small, under 1 second. With a fixed number of sinks, the delay tends to go up as the number of source increase. For example, when the the network is configured with 2 sinks, the delay goes from 55ms, 325ms, to 556ms as the number of sources increase from 4, 6, to 8.

3.7 Dual-Sink

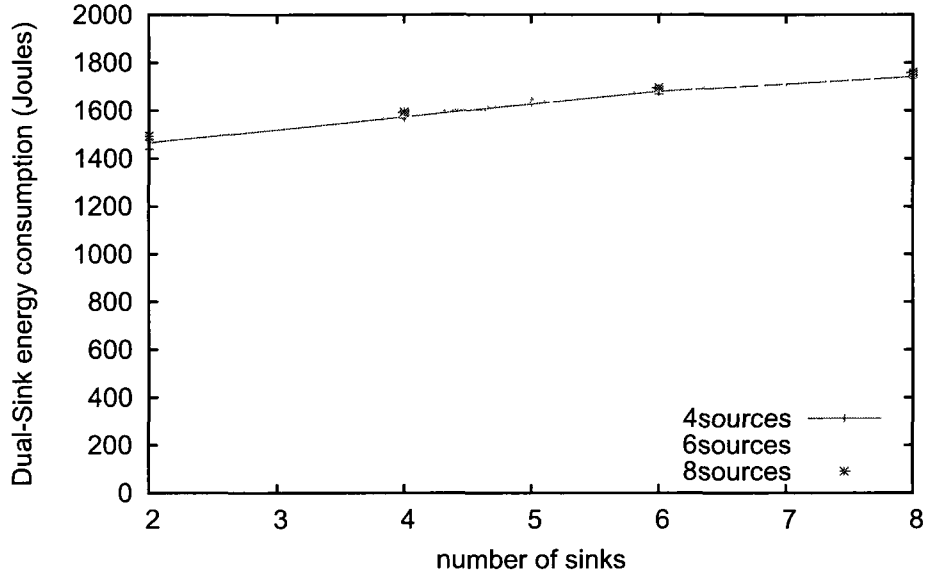


Figure 3.16: Dual-Sink energy consumption vs. numbers of sinks and sources

Figure 3.16 shows the energy consumption of Dual-Sink. The energy consumption goes up linearly as the number of sinks increase. A greater number of sinks leads to a greater number of routing trees, which causes more energy consumption.

The change of the number of sources has little effect on the energy consumption. For example when the network is configured with 4 sinks, the energy goes up from 1573, 1581, to 1594 as the number of sources increase from 4, 6, to 8. In Dual-Sink there is no grid-like infrastructure for each source so more sources do not introduce more energy consumption.

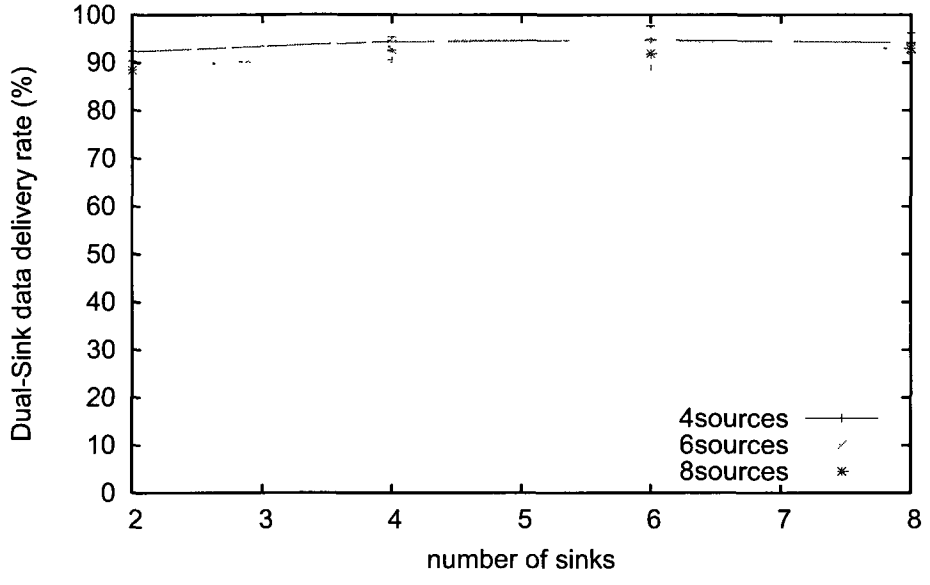


Figure 3.17: Dual-Sink data delivery rate vs. numbers of sinks and sources

Figure 3.17 shows the data delivery ratio of Dual-Sink. The change of the data delivery ratio is relatively small as the number of sinks change. For instance, in a network with 8 sources, the data delivery ratio increases from about 88% to 93% when the number of sinks increases from 2 to 8. If a sensor node cannot find a route to a mobile sink, it can always deliver its data to the static sink.

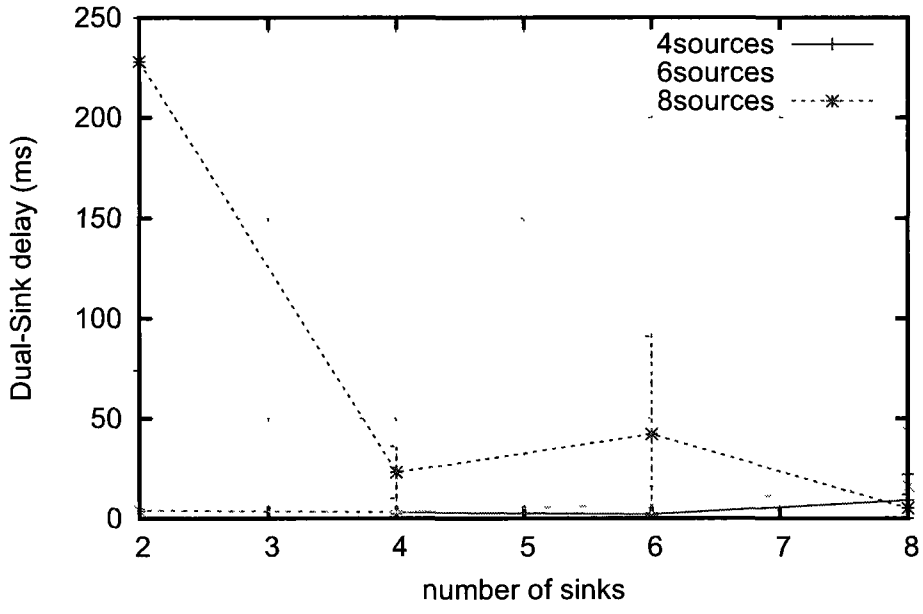


Figure 3.18: Dual-Sink delay vs. numbers of sinks and sources

Figure 3.18 shows the delay of Dual-Sink. The delay is very small, for almost all network configurations it is under 50ms.

Chapter 4

Comparison Analysis

This chapter compares the results of the simulation tests of six protocols, analyzes the underlying reasons and points out the performance issues.

4.1 Energy consumption and data delivery ratio

At a glance

Under the same simulation setting, the energy consumption of the six protocols can be categorized into three classes. TTDD is in the high class, as it has much higher energy consumption than the other five protocols. The grid infrastructure on a per-source basis in TTDD consumes a great deal of energy while the other five protocols have no infrastructure cost. Locator is in the middle class; it is lower than TTDD but higher than the other four protocols. Dual-Sink, Efficient Routing, LURP and Twin-Route are in the low class. All of these protocols have some restriction in the broadcasting of their routing update or location update message. Table 4.1 shows the summary of the energy consumption for each protocol. Figure 4.1 presents a more intuitive observation of the protocols.

Table 4.1: Summary of the energy consumption for each protocol

Class	Energy consumption	Range
High	TTDD	2700J - 7000J
Middle	Locator	1500J – 2300J
Low	Dual-Sink	1500J – 1800J
Low	LURP	1500J - 1700J
Low	Efficient Routing	1500J - 1700J
Low	Twin-Route	1400J - 1500J

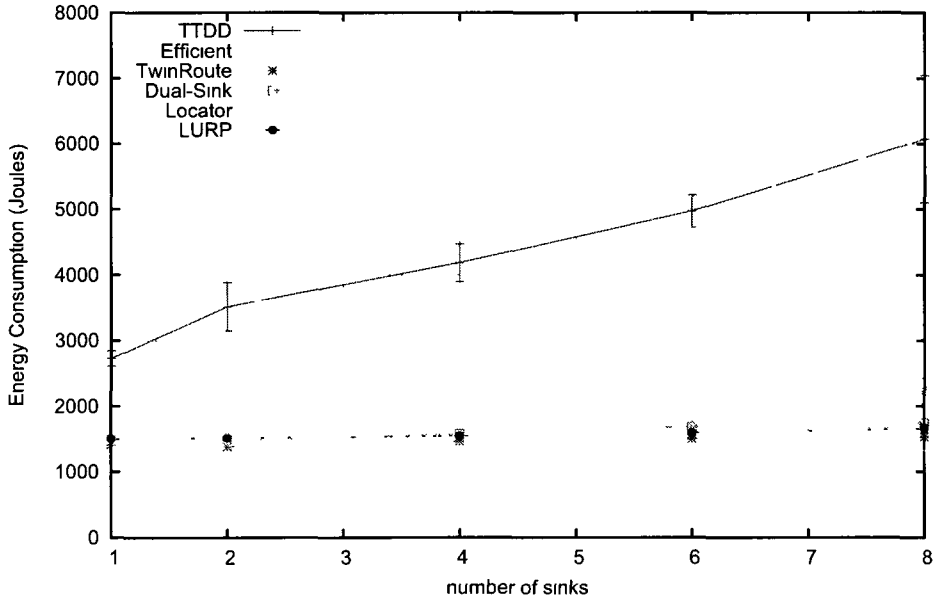


Figure 4.1: Energy consumption vs number of sinks with 200 sensors including 6 sources

Under the same simulation setting, the data delivery rate of the six protocols can also be categorized into three classes. Efficient Routing and TTDD have almost the same high performance in their data delivery ability, except that when the network is configured with one sink Efficient Routing performs better than TTDD. Dual-Sink and TwinRoute

share some similarity in that both sensors may have two options when delivering data. However, Dual-Sink has better performance than TwinRoute. LURP and Locator have relatively poor performance in their data delivery ability, with Locator having the worst performance. Table 4.2 shows the summary of the data delivery rate for each protocol. Figure 4.2 presents a more intuitive observation of the protocols.

Table 4.2: Summary of the data delivery rate for each protocol

Class	Delivery rate	Range
High	Efficient Routing	88% – 98%
High	TTDD	70% – 97%
Middle	Dual-Sink	92% – 94%
Middle	Twin-Route	75% – 90%
Low	LURP	40% – 90%
Low	Locator	15% – 80%

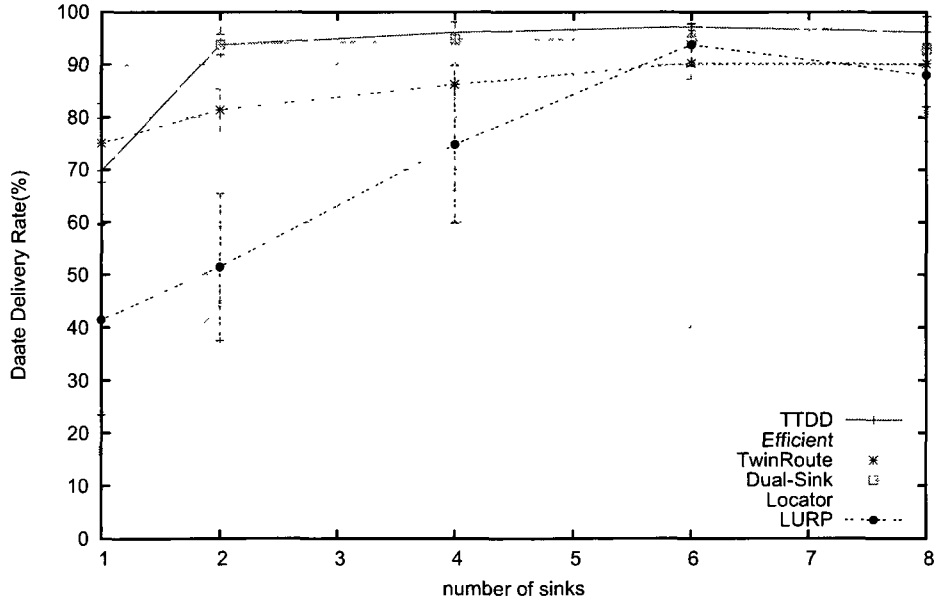


Figure 4.2: Delivery ratio vs number of sinks with 200 sensors including 6 sources

TTDD

TTDD has the highest energy consumption among all six protocols due to the unique data collection mode employed by TTDD. Instead of waiting for a location update, a source node builds a grid infrastructure proactively. A great amount of energy is spent on grid construction. In its implementation Data Announcement packets are sent by sources to launch the grid construction. Whenever a sensor node becomes a data dissemination node on the grid, it broadcasts DN Declare packets to its neighboring sensor nodes. Because each source node builds its own grid, and each grid uses different sets of sensor nodes as data dissemination nodes, the number of DN Declare packets sent by data dissemination nodes will increase in proportion with the number of source nodes in the network.

Upstream Update packets are sent periodically by data dissemination nodes in order to forward queries from the sinks and receive data from the sources continuously. Again, because the grid is built on a per-source basis, the number of dissemination nodes will increase in proportion with the number of sources in the networks. Therefore a significant amount of energy is consumed by a large quantity of Upstream Update packets sent by the dissemination nodes. Also, part of the energy has been spent on grid maintenance. A mechanism called upstream information duplication, which accommodates formation of new dissemination nodes, is used to handle possible dissemination node failure, while the other five protocols have no explicit fault tolerance mechanism to support the network robustness.

At the expense of having the highest energy consumption, TTDD achieves a better data delivery rate than four of the protocols, and performs almost equally well as Efficient Routing protocol. Query fowarding is TTDD's unique method for the source to find the path to the sink, promising that the path is correct which results in high data delivery rate. In other protocols, such as Locator and LURP, sensors may use out-of-date sink

location information when forwarding data to sinks, resulting in a low delivery rate.

Locator

Locator does not perform well either in energy consumption or in data delivery rate. It consumes more energy than the other four protocols except TTDD, and it has the poorest data delivery performance among all six protocols.

Locators are just normal sensor nodes but they know a sink's current position and reply location query from the sensors. In the implementation, sensor nodes are selected to be locators given that they are closest to the geographic hash table's hashed positions. A sink sends a location update packet to locators periodically. Locator has specified that a sink will not send a location update packet if its position change is less than a specific minimal distance, unless the previous location update is expired. However, it has no effective mechanism to restrict the range of broadcasting of location update packets, therefore its energy consumption is higher than LURP, Efficient Routing, DualSink and TwinRoute.

Another reason is that the fundamental data forwarding mechanism used by Locator is simple greedy geographic forwarding. Because a sensor node forwards data packet to its neighboring node which has the smallest distance to the destination node, a sensor must know its neighbor's position. In the implementation, sensors exchange hello packets periodically to set up and maintain a neighbor nodes table which contains neighbor nodes' positions. However, information exchange between neighboring sensors does not exist in topology-based routing protocols like Efficient Routing, TwinRoute and Dual-Sink. Inevitably, these periodical hello packets consume an additional amount of energy.

When a source node has data to report, it sends location query packets to locators. After receiving the location reply packet from a locator, using acquired sink position

information a source node can deliver its sensed data to the sink using simple greedy geographic forwarding. A performance issue is that if a sink moves relatively faster, forwarding data packets from relatively far-away source to the sink is most likely to end in failure because the sink has already moved out of the previous position. Sensors may still use out-of-date sink location information, which was acquired from locators, resulting in low delivery rate, unless a sink sends location updates more frequently when it is moving faster. However, doing so will cause more energy consumption of sensor nodes.

LURP

The energy consumption and data delivery rate of LURP are in the low class.

LURP divides the network into different geographic areas with different area identifier. A sink broadcasts a periodical location update packet. The packet has a cross area flag which can be turned on when a sink detects that it has moved out of the previous area by checking its current position. If this flag is set, the location update packet, which contains the sink's current area identifier will be flooded throughout the senor field so that every sensor node knows in which area the sink is currrently visiting. If a sink is just moving inside one area, it turns off the cross area flag in its location update packet. The packets will only be propagated in the sink's current area, so only sensors in the same area with the sink will get updated with the current position of the sink.

The above mechanism can reduce overall network energy consumption to some degree because the propagation of the location update packet is restricted within one area when a sink is moving inside the area. This restriction is why LURP has lower energy consumption than Locator, which is also a location-based routing protocol. However, the size of the area needs to be carefully selected. If it is too small, sink's cross area location update will be frequently flooded throughout the sensor field when a sink moves from one small

area to another small area. If it is too big, then the cost of the location update within the local area will be too much. The size of the area should be given an appropriate value to save energy consumption. Like Locator protocol, the data forwarding mechanism used by LURP is simple greedy geographic forwarding. Periodically exchanging hello packets between neighbor sensors is needed to maintain the neighbor table, thus the amount of energy used cannot be avoided.

Because each sensor is informed of the sink's area identifier, when a source has data to report, the data will first be forwarded to the sink's current area. Once data arrives in the sink's area, it will be forwarded to the sink by sensors which know the sink's current position. However, there is a problem in data delivery. When a sink is leaving an area, the location update packet with crossarea flag turned on will be flooded throughout the sensor field so that every sensor will know sink's new area identifier. If we take propagation delay into consideration, data that is being sent to the previous area of the sink will get lost easily if the forwarding nodes have not yet received the sink's latest location update packet. Sending a location update more frequently will offset this problem, but frequent location updates will lead to more energy consumption of sensor nodes.

Although LURP and Locator are both location-based routing protocols, Locator is worse than LURP in its data delivery ability. Due to the design of the Locator protocol, source nodes do not receive location update packets directly, they have to query locators to get sink location information. This extra step makes the location information more likely to be out of date, which leads to the lower data delivery ratio.

Efficient Routing

Efficient Routing shows the best data delivery ability and has the lower energy consumption.

In Efficient Routing, a sink broadcasts routing update packet periodically, and accordingly the routing tree rooted at the sink gets updated periodically. Each sensor node stores its distance to the sink in number of hops and knows its next hop neighbor sensor node in the direction towards the sink. The routing update packet has a hop value attribute that indicates the distance to the sink. After receiving the routing update packet, a sensor node computes the distance change to the sink. When the change in its distance to the sink exceeds a given percentage threshold, the sensor node will modify its routing table and broadcast the routing update packet further, otherwise it simply drops off the packet.

From the above algorithm we can see that the key tunable parameter is threshold percent, which determines the propagation range of the routing update packet. The effect of threshold percent is that sensor nodes far away from the sink update their cost and route to the sink less frequently than these sensors near the sink, demonstrated in Figure 2.8. Because not every sensor node re-transmits the routing update packet every time it receives the routing update packet, so a large amount of energy can be saved. On the other hand, sensors in the neighborhood of sinks are always updated immediately, and they are responsible for the final data forwarding, so when data arrives, it can be correctly forwarded to the sink.

TwinRoute

TwinRoute has the lowest energy consumption among the six protocols due to its effective multiple restrictions for construction of the P-scheme and R-scheme routing trees. With a R-scheme tree, the energy consumed on building and updating the tree is relatively small. The tree is built in reaction to sink's random movement, its depth is controlled by the parameter tree depth and the route get demolished after a short time interval set by a timer. Every sensor node can attach itself to a P-scheme tree, but not every sensor node is allowed to associate itself with a R-scheme tree. The parameter scheme

preference is used to determine whether or not a node can become part of R-scheme tree. With a P-scheme tree, based on the statistic information about sink visiting history, a sensor node can become a storage node. Once a storage node is chosen, relatively stable routes are used from sensor nodes to the storage node, thus the energy spent on updating a P-scheme routing tree is very small.

The data delivery rate of TwinRoute is in the middle class. A stable P-scheme routing tree is always ready for a sensor node to forward its data. When a sink passes by, an instant small R-scheme tree is formed, so a sensor node will have the chance to offload its data directly to the visiting sink along the R-scheme tree. There are two concerns with TwinRoute. First, The rule which determines if a node can become part of the R-scheme tree is that the senor node's distance to root node in the R-scheme tree must be shorter than its distance to the storage node in the P-scheme tree by the parameter scheme preference. It is beneficial for these sensor nodes which are many hops away from the storage node, because they are more likely to be allowed to take part in the R-scheme tree, demonstrated in Figure 2.10, while sensors near the storage node may only have one data delivery option. Second, the P-scheme routing tree cannot guarantee that the data is delivered to the sink. For example, if a sensor node has data to report, but at that moment there is no R-scheme tree for it to take advantage of, it has to forward its data to the storage node of the P-scheme tree. Data is buffered in the storage node waiting for a sink which is supposed to come according to its visiting records, however if the sink does not appear, which is highly likely to happen in the real application environment, then the data is missed.

Dual-Sink

The energy consumption of Dual-Sink is in the low class and data delivery rate of it is in the middle class.

There are two types of sinks in the Dual-Sink network, static sinks and mobile sinks respectively. At the start of network operation, the static sink broadcasts a hello packet which is flooded throughout the sensor field. The broadcasting of a static sink hello packet only happens once, so its energy consumption is affordable. A mobile sink broadcasts a hello packet to its neighbors periodically. This hello packet is not aimed to be flooded in the network. In order to restrict the broadcast range of the hello packet, the hello packet is configured with a ttl value. The mobile sink sends the hello packet with its initial ttl value set to a predefined value. When a sensor receives the sink's hello packet, it will reduce the ttl value by one. If the reduced ttl value equals zero, the sensor will not continue to broadcast the sink's hello packet. As not all of the sensor nodes receive the hello packet originating from the mobile sink, only some sensor nodes in the network form a relatively small routing tree, so the overall energy consumption is under control.

Just like TwinRoute, a sensor node may have two choices when it delivers data. It has a stable route toward the static sink, but it could also have the opportunity to forward its data to the moving sink if it could receive the hello packet originating from the moving sink. However, the delivery ratio in Dual-Sink is better than TwinRoute. There are two reasons. First, in TwinRoute, when data is delivered to the storage node, which is the root node of the P-scheme tree, it is possible that the data may not be collected when a mobile sink does not visit the storage node as expected. In Dual-Sink, there is a guaranteed route to the static sink if data has no chance to be forwarded to the mobile sink. Second, there is no other restrictions for a sensor node to be part of a routing tree surrounding a mobile sink as long as the ttl value permits.

There is an issue in the design of Dual-Sink. Obviously sensors surrounding the static sink use up their energy more quickly because they are the basic choices if sensors cannot send data to mobile sinks. The protocol does not provide a solution to deal with this situation.

4.2 Network lifetime

The network lifetime [52, 53]of a WSN depends on the energy of each individual sensor. In the simulation tests, the initial energy level for each sensor was set to a small number, to mimic the real application environment where sensors only have limited battery power. At the beginning of the simulation tests there were 200 sensors. The tests were run to show how many sensors of each protocol were dead at the end of the simulation. The number of mobile sinks varied from 2, 4, 6, to 8 to evaluate the impact on the network performance of the six protocols. The simulation parameters used in the tests are listed in Table 4.3.

Table 4.3: Summary of the simulation parameters of network lifetime

Parameter	value
Simulation time	200s
Simulation area	2000mx2000m
Number of sensor nodes	200
Number of sinks	2,4,6,8
Number of source sensor nodes	6
Data rate of source sensor nodes	1pkt/s
Speed of mobile sink	10 m/s
TX power dissipation rate	0.66w
Rx power dissipation rate	0.395w
Idle power dissipation rate	0.035w
Initial energy of sensor node	7J

Figures 4.3, 4.4, 4.5, and 4.6 show the test results for each protocol when the network is configured with 2, 4, 6 and 8 sinks and there are 6 source nodes out of 200 sensor nodes. Note for Dual-Sink, there is always one static sink in each network configuration.

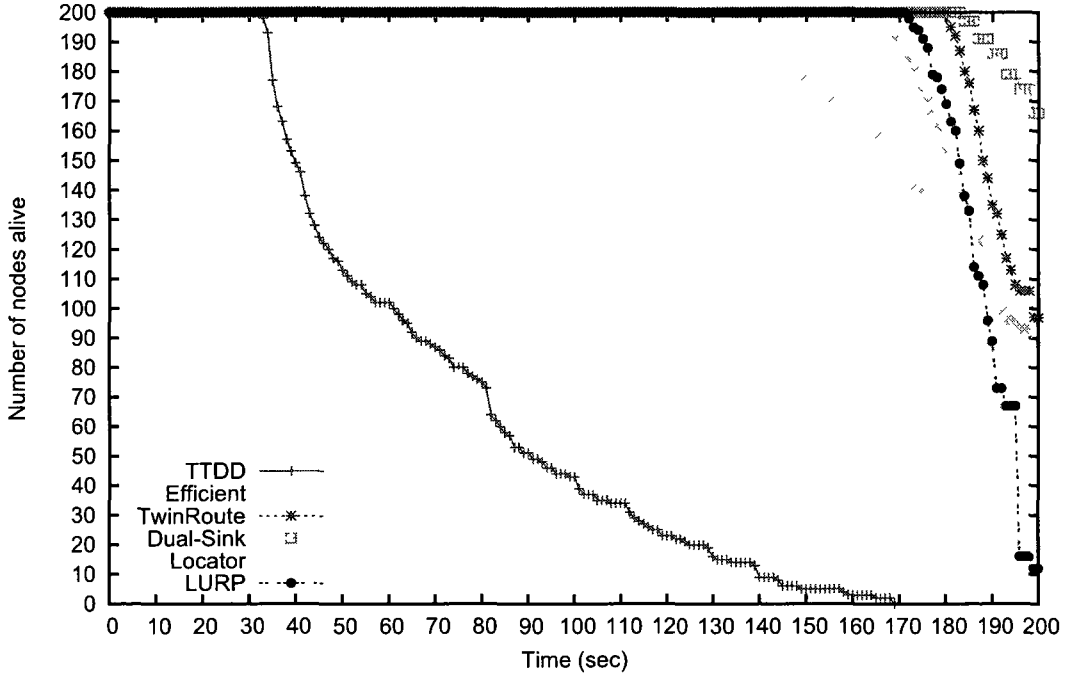


Figure 4.3: The network lifetime for each protocol when configured with 2 sinks

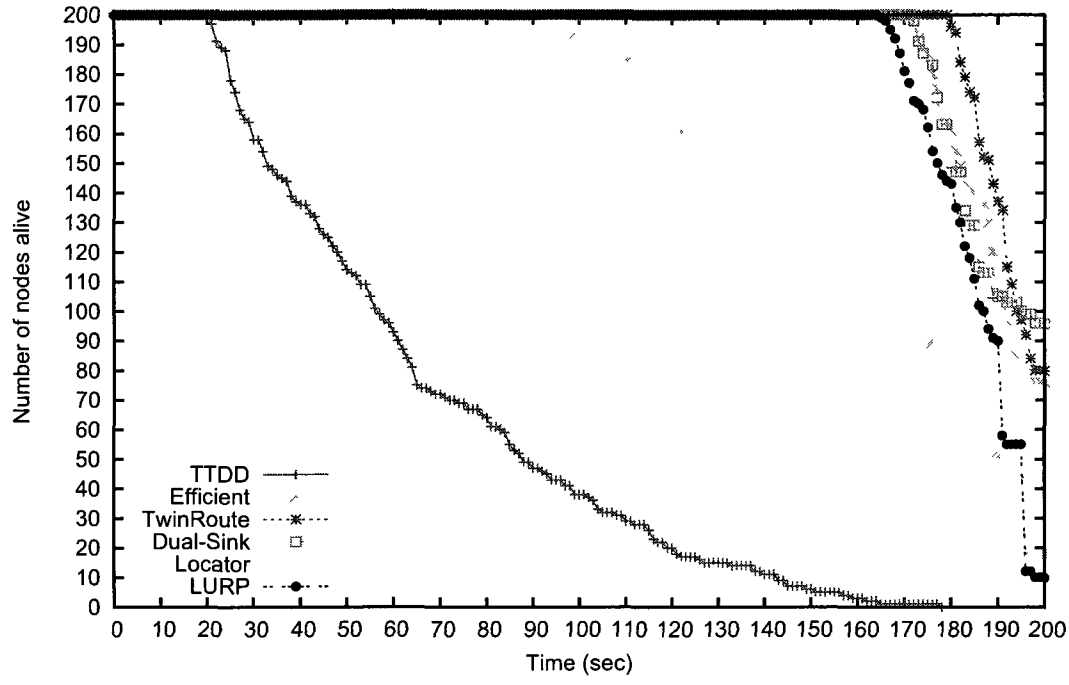


Figure 4.4: The network lifetime for each protocol when configured with 4 sinks

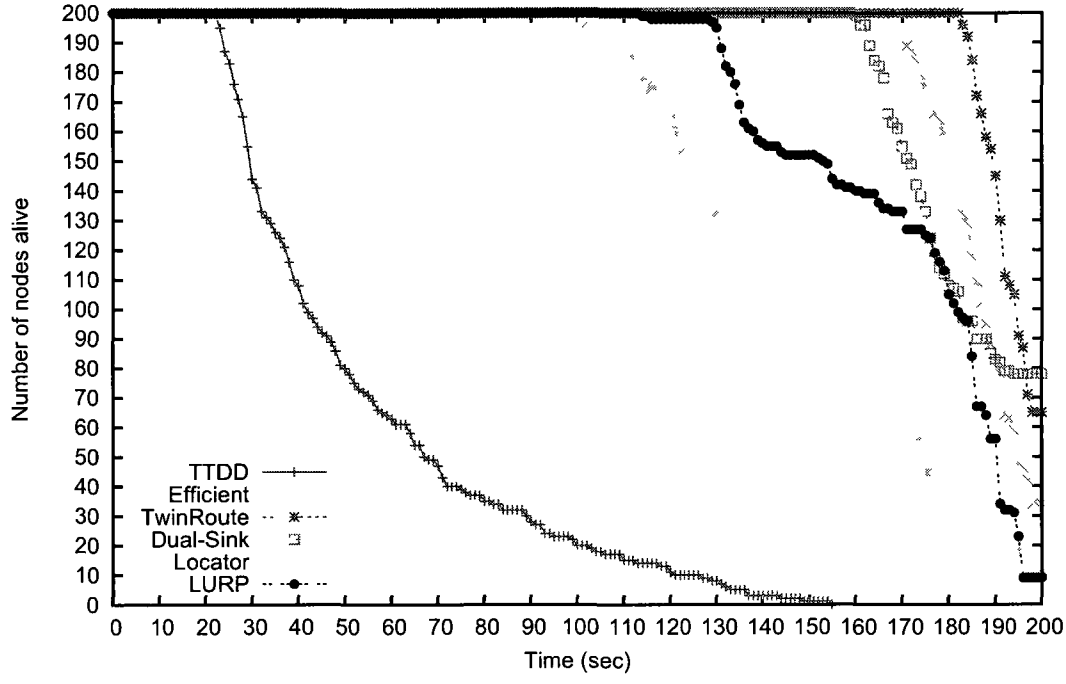


Figure 4.5: The network lifetime for each protocol when configured with 6 sinks

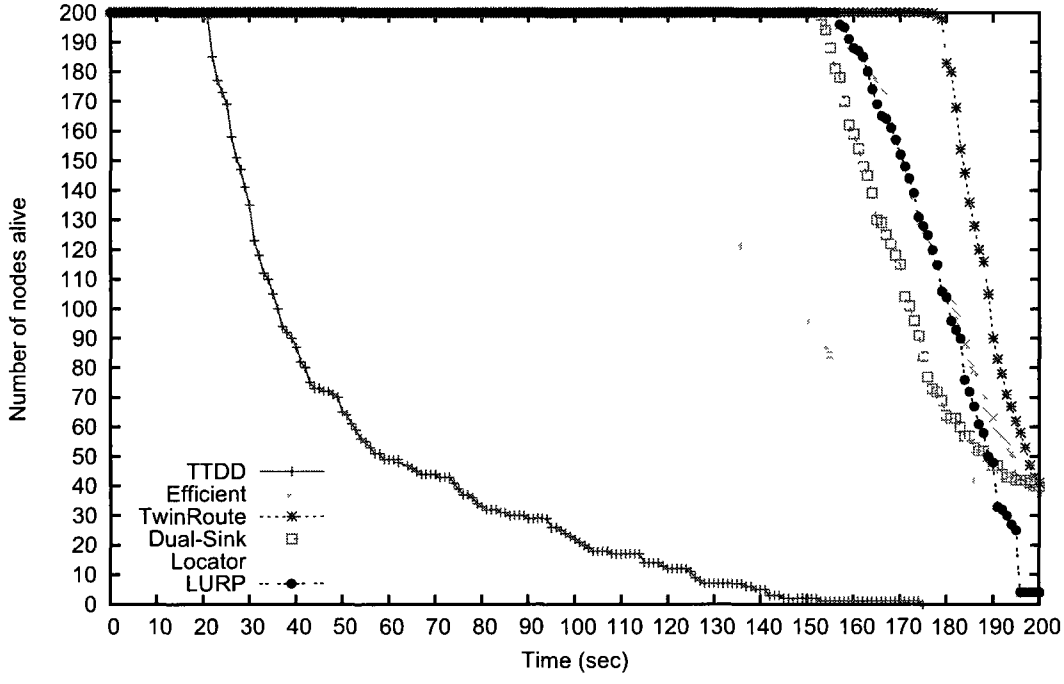


Figure 4.6: The network lifetime for each protocol when configured with 8 sinks

The above four graphs show that in TTDD the first sensor is dead at about 20 to 30 seconds and all sensors are dead before the simulation time is finished, proving that TTDD has the highest energy consumption and conforms to the result of the previous performance evaluation test.

On average, in Locator the first sensor is dead at about 100 seconds and most of the sensors are dead at the end of simulation time. This proves that Locator has the second highest energy consumption and conforms to the result of the previous performance evaluation test.

Generally speaking, for Efficient Routing, Dual-Sink and LURP, the first sensors are dead at about 150 to 180 seconds, and for Twin-Route, the first sensor is dead at about 180 seconds. With the number of sinks increasing, more sensors are dead at the end of the

simulation time. This proves that Efficient Routing, LURP, Dual-Sink and Twin-Route have better energy efficiency than TTDD and Locator, and conforms to the results of the previous performance evaluation tests.

4.3 Delay

Generally speaking, all protocols except Locator show good performance in delay. As shown in Figure 4.7, the delays are under 1 second for TTDD, LURP, Efficient Routing, TwinRoute and Dual-Sink. Specifically, the delays of TTDD and TwinRoute are hundreds of mili-seconds, the delays of LURP and Dual-Sink are under 50 ms, and the delay of Efficient Routing is under 50 ms when the network is configured with more than 1 sink.

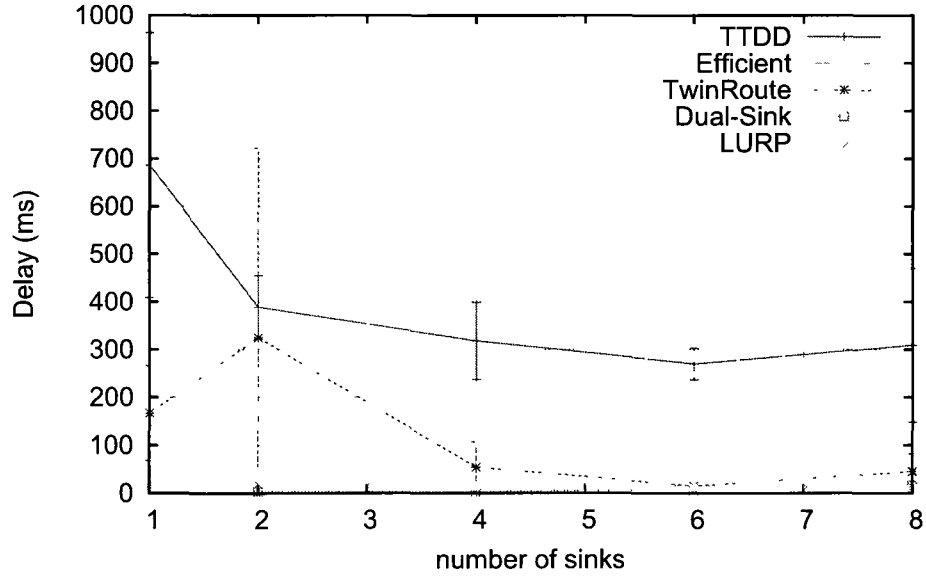


Figure 4.7: Delay rate vs number of sinks with 200 sensors including 6 sources

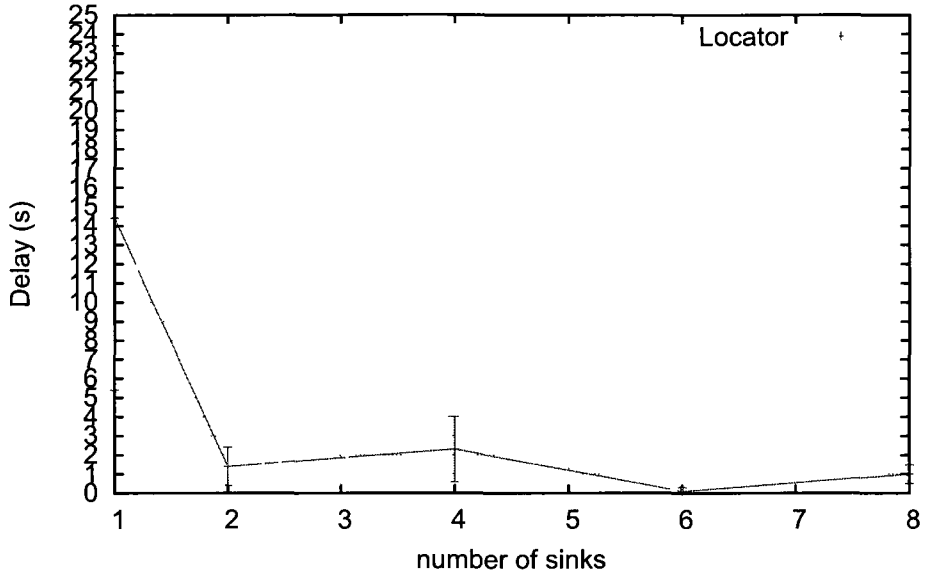


Figure 4.8: Locator delay vs number of sinks with 200 sensors including 6 sources

However for Locator, the delay is several seconds, see Figure 4.8. The reason for this is sensors get the sink location information from locators, and this extra step causes a large delay.

Chapter 5

Conclusion and future work

This thesis provides a comprehensive study of six routing protocols, including detailed algorithm descriptions, performance evaluations, and comparison analysis. In this chapter, a summary of the comparison of the six protocols is provided, followed by the possible future research.

5.1 Conclusion

The most important task in designing a data routing algorithm in a wireless sensor network is minimizing the energy consumption of sensors to prolong the network lifetime while achieving desirable network performance such as the data delivery rate. To this end, TTDD, Locator, LURP, Efficient Routing, TwinRoute and Dual-Sink all developed their own solutions, all of which support sink random movement.

These six protocols can be categorized into two classes, topology-based routing and location-based routing. Efficient Routing, TwinRoute and Dual-Sink are topology-based routing protocols in which sensors and sinks have no awareness of their locations. TTDD, Locator and LURP are location-based routing protocols in which the position information of sensors and sinks are exploited to route data to sinks.

TTDD is a proactive routing protocol in the sense that a source proactively builds a grid before data delivery. Locator, LURP, Efficient Routing and Dual-sink are reactive routing protocols in which sensors wait for either location updates or routing updates originating from sinks. TwinRoute adopts a hybrid approach which combines proactive and reactive schemes; it builds long-term stable routing trees beforehand and short-term instant trees in reaction to sink's movement.

The results of the performance evaluation show that, under the same simulation setting, the three topology-based routing protocols have better overall network performance than the three location-based protocols. The most important thing that distinguishes TTDD from the other five protocols is its unique grid infrastructure, which has an advantage and a disadvantage. TTDD shows very good data delivery ability, while the grid infrastructure has a built-in mechanism to support network robustness which the other five protocols lack. However, a grid infrastructure leads to high energy consumption, which is why TTDD has a much higher energy consumption than the other five protocols. LURP and Locator have much lower energy consumption than TTDD, but their data delivery ability is inferior to other protocols. In order to improve the data delivery rate, more frequent location updates are needed which will cause more energy consumption. Topology-based protocols have low energy consumption while not compromising network performance such as the data delivery rate. TwinRoute has the lowest energy consumption because it has a more complex control mechanism than Efficient Routing and Dual-Sink, which only use a single parameter to limit the broadcast range of routing update packets. Efficient Routing shows very good data delivery ability because it does not take chances like Dual-Sink and TwinRoute. Table 5.1 summarizes the comparison of the six protocols.

Table 5.1: Summary of the comparison of the six routing protocols in WSNs

Protocols	classification	Proactive	Reactive	Fault tolerance	Energy consumption	Data delivery rate
TTDD	Location-based	Yes	No	Yes	High	High
Locator	Location-based	No	Yes	No	Middle	Low
LURP	Location-based	No	Yes	No	Low	Low
Efficient	Topology-based	No	Yes	No	Low	High
TwinRoute	Topology-based	Yes	Yes	No	Low	Middle
Dual-Sink	Topology-based	No	Yes	No	Low	Middle

5.2 Future work

Possible research in the future is proposed as follows.

- This thesis investigates six protocols which support sink random movement. However, sinks can also move on fixed or controlled paths in WSNs. In order to study the impact of different sink mobility patterns on network performance of WSNs, we plan to investigate several other data routing protocols such as [24–28], which use controlled sink mobility. The selected protocols will be implemented in ns-2 and evaluated under the same simulation setting as the six protocols in this thesis, so the results can be compared and analyzed.

- TwinRoute is a promising data routing protocol. First, it has the lowest energy consumption among the six protocols. Second, it is a combination of proactive and reactive schemes which can be easily adapted to many applications. However, the mixture of two schemes is not flexible and there is no mechanism in TwinRoute to cope with sensor failure. Inspired by TwinRoute, we plan to design a new data routing protocol for a wireless sensor network with mobile sinks, which can flexibly mix multiple routing schemes together to achieve a better data delivery rate. Furthermore, we will add detect and recovery functions to assure network reliability.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, A survey on sensor networks, IEEE Communications Magazine, 40(8):102-114, 2002.

[2] R.Kay and F.Mattern, The design space of wireless sensor networks. IEEE Wireless Communications, 11(6):54-61, 2004.

[3] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, John Anderson, Wireless sensor networks for habitat monitoring, Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, Pages:88-97, 2002

[4] Aleksandar Milenkovic, Chris Otto, Wireless sensor networks for personal health monitoring: Issues and an implementation, Computer Communications, Vol.29, Issues 13-14, Pages:2521-2533, 2006.

[5] E.M. Petriu, N.D. Georganas, D.C. Petriu, D. Makrakis,V.Z. Groza, Sensor-based information appliances, IEEE Instrumentation and Measurement Magazine, 3(4):31-35, 2000.

[6] wildlife, http://www.wildcru.org/wytham/about.php

[7] Jamal N. Al-Karaki and Ahmed E. Kamal, Routing techniques in wireless sensor networks: a survey. Wireless Communications, IEEE 11(6):6-28, 2004.

[8] Joel K. Young, A Practical Guide to Battery Technologies for Wireless Sensor Networking, http://www.sensorsmag.com/networking-communications/batteries

[9] R.A. Abd-Alhameed, K.V. Horoshenkov, Y.F. Hu, C.H. See, D. Zhou, Measure The Range Of Sensor Networks, http://www.mwrf.com/Articles/Index.cfm?Ad=1&ArticleID=19915, 2008.

[10] C. Intanagonwiwat, R. Govindan and D. Estrin, Directed diffusion: a scalable and robust communication paradigm for sensor networks, Proceedings of ACM MobiCom, Pages:56-67, 2000.

[11] Y. Yao and J. Gehrke, The cougar approach to in-network query processing in sensor networks, in SIGMOD Record, 31(3):9-18, 2002.

[12] D. Braginsky and D. Estrin, Rumor Routing Algorithm for Sensor Networks, in the Proceedings of the First Workshop on Sensor Networks and Applications (WSNA), Pages:22-31, 2002.

[13] C. Schurgers and M.B. Srivastava, Energy efficient routing in wireless sensor networks, in Proceedings of IEEE MILCOM Pages:357-361, 2001.

[14] M. Chu, H. Haussecker, and F. Zhao, Scalable Information-Driven Sensor Querying and Routing for ad hoc Heteroge-neous Sensor Networks, The International Journal of High Performance Computing Applications, 16(3):293-313, 2002.

[15] F. Ye, A. Chen, S. Liu, L. Zhang, A scalable solution to minimum cost forwarding in large sensor networks, Proceedings of the tenth International Conference on Computer Communications and Networks (ICCCN), Pages:304-309, 2001.

[16] R. C. Shah and J. Rabaey, Energy Aware Routing for Low Energy Ad Hoc Sensor Networks, IEEE Wireless Communications and Networking Conference (WCNC), Pages:350-355 vol.1, 2002.

[17] N. Sadagopan et al., The ACQUIRE mechanism for efficient querying in sensor networks, in the Proceedings of the First IEEE International Workshop on Sensor Network Protocol and Applications, Pages:149-155, 2003.

[18] W. Heinzelman, A. Chandrakasan and H. Balakrishnan, Energy-Effcient Communication Protocol for Wireless Microsensor Networks, Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS '00), 10pp, vol.2, 2000.

[19] A. Manjeshwar and D. P. Agarwal, TEEN: a routing protocol for enhanced effciency in wireless sensor networks, Proceedings of the 15th International Parallel and Distributed Processing Symposium. Pages:2009-2015, 2001.

[20] A. Manjeshwar and D. P. Agarwal, APTEEN: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks, Proceedings of the 16th International Parallel and Distributed Processing Symposium, Pages:195-202, 2002.

[21] S. Lindsey, C. Raghavendra, PEGASIS: Power-Efficient Gathering in Sensor Information Systems, IEEE Aerospace Conference Proceedings, Vol.3, Pages:1125-1130, 2002.

[22] Jamal N. Al-Karaki, Raza Ul-Mustafa, Ahmed E. Kamal, Data Aggregation in Wireless Sensor Networks - Exact and Approximate Algorithms, Proceedings of IEEE Workshop on High Performance Switching and Routing (HPSR), Pages:241-245, 2004.

[23] Jung Hyun Jun, Bin Xie, and Dharma P. Agrawal, Chapter 24:Wireless Mobile Sensor Networks: Protocols and Mobility Strategies, Guide to Wireless Sensor Networks, Computer Communications and Networks, 2009.

[24] S. Gandham, M. Dawande, R. Prakash, and S. Venkatesan, Energy efficient schemes

for wireless sensor networks with multiple mobile base stations, in Global Telecommunications Conference(GlobeCom), Vol.1, Pages:377-381, 2003.

[25] Z. Wang, S. Basagni, E. Melachrinoudis, and C. Petrioli, Exploiting sink mobility for maximizing sensor networks lifetime, in Proceeding of the 38th Hawaii International Conference on System Sciences, Pages:287-295, 2005.

[26] J. Luo and J.P. Hubaux, Joint mobility and routing for lifetime elongation in wireless sensor networks, in Intl Conference on Computer Communications (Infocom), Pages:1735C1746, 2005.

[27] J. Luo, J. Panchard, M. Piorkowski, M. Grossglauser and J.P. Hubaux, MobiRoute: Routing towards a Mobile Sink for Improving Lifetime in Sensor Networks, in Proceeding of 2nd IEEE/ACM International Conference on Distributed Computing in Sensor Systems(DCOSS '06), Pages:480-497, 2006.

[28] Stefano Basagni, Alessio Carosi, Emanuel Melachrinoudis, Chiara Petrioli and Z. Maria Wang, Controlled sink mobility for prolonging wireless sensor networks lifetime, Wireless Networks, 14(6): 831-858, 2008.

[29] Hyung Seok Kim, Tarek F. Abdelzaher, Wook Hyun Kwon, Minimum-Energy Asynchronous Dissemination to Mobile Sinks in Wireless Sensor Networks, in Proceeding of the 1st international conference on Embedded Networked Sensor Systems, Pages:193-204, 2003.

[30] P. Baruah, R. Urgaonkar and B. Krishnamachari, Learning-Enforced Time Domain Routing to Mobile Sinks in Wireless Sensor Fields, in Proceeding of 29th Annual IEEE International Conference on Local Computer Networks, pages:525-532, 2004.

[31] A.A. Somasundara, A. Kansal, D.D. Jea, Jea, D.Estrin, and M. B. Srivastava, Controllably Mobile Infrastructure for Low Energy Embedded Networks, in IEEE Transactions on Mobile Computing, 5(8):958-973, 2006.

[32] D. Jea, A. A. Somasundara, M. B. Srivastava, Multiple Controlled Mobile Elements (Data Mules) for Data Collection in Sensor Networks, in Proceeding of 1st IEEE/ACM International Conference on Distributed Computing in Sensor Systems (DCOSS '05), Pages:244-257, 2005.

[33] Z. Vincze, D. Vass, R. Vida and A. Vidacs, A. Telcs, Adaptive sink mobility in event-driven multi-hop wireless sensor networks, in Proceedings of the First International Conference on Integrated Internet ad hoc and sensor network, Article No.:13, 2006.

[34] Wei Wang, Vikram Srinivasan and Kee-Chaing Chua, Using mobile relays to prolong the lifetime of wireless sensor networks, in Proceedings of the 11th Annual International Conference on Mobile computing and networking, Pages:270-283, 2005.

[35] H. Luo, F. Ye, J. Cheng, S. Lu, and L. Zhang, TTDD: Two-tier Data Dissemination in large-scale wireless sensor networks. Wireless Networks, Vol.11, Pages:161-175, 2005.

[36] Gyudong Shim and Daeyeon Park, Locators of Mobile Sinks for Wireless Sensor Networks. Proceedings of the 2006 International Conference on Parallel Processing Workshops (ICPPW'06) Pages:159-164, 2006.

[37] Guojun Wang, Tian Wang, Weijia Jia, Minyi Guo, Hsiao-Hwa Chen and Mohsen Guizan, Local Update-Based Routing Protocol in Wireless Sensor Networks with Mobile Sinks. IEEE International Conference on Communications, Pages:3094-3099, 2007.

[38] Kristof Fodor and Attila Vidacs, Efficient Routing to Mobile Sinks in Wireless Sensor Networks. Proceedings of the 3rd international conference on Wireless internet, Article No.:32, 2007.

[39] Xiaobing Wu and Guihai Chen, DualSink: Using Mobile and Static Sinks for Lifetime Improvement in Wireless Sensor Networks. Proceedings of Sixteenth Interna-

tional Conference on Computer Communications and Networks, Pages:1297-1302, 2007.

[40] Ricklef Wohlers, Niki Trigoni, Rui Zhang and Stephen Ellwood, TwinRoute: Energy-Efficient Data Collection in Fixed Sensor Networks with Mobile Sinks. Tenth International Conference on Mobile Data Management: Systems, Services and Middleware, Pages:192-201, 2009.

[41] VINT project, The Network Simulator ns2. http://www.isi.edu/nsnam/ns/.

[42] P. Levis, N. Lee, M. Welsh, and D. Culler, Tossim: accurate and scalable simulation of entire tinyos applications. Proceedings of the 1st international conference on Embedded networked sensor systems, Pages:126-137, 2003.

[43] The MathWorks, Inc., http://www.mathworks.com/.

[44] B. Karp and H. T. Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking, Pages:243 - 254, 2000.

[45] J. Albowicz, A. Chen and L. Zhang, Recursive position estimation in sensor networks. In Proceedings of Ninth International Conference on Network Protocols, pages:35-41, 2001.

[46] A. Savvides, C. Han, M. Srivastava, Dynamic fine-grained localization in ad-hoc networks of sensors, Proceedings of ACM MobiCom pp.166-179, 2001.

[47] J. Hightower and G. Borriello, Location systems for ubiquitous computing,IEEE Computer Magazine 34(8):57-66, 2001.

[48] Sylvia Ratnasamy, Brad Karp, Li Yin, Fang Yu, Deborah Estrin, Ramesh Govindan, Scott Shenker, GHT: a geographic hash table for data-centric storage, Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, Pages:78-87, 2002.

[49] Christian Bettstetter, Giovanni Resta, and Paolo Santi. The node distribution of the random waypoint mobility model for wireless ad hoc networks. IEEE Transactions on Mobile Computing, 2(3):257-269, 2003.

[50] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang and M. Gerla, The impact of multihop wireless channel on TCP throughput and loss. Proceedings of International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM03), Pages:1744-1753 vol.3, 2003.

[51] B. Bensaou, Y. Wang, and C. C. Ko. Fair medium access in 802.11 based wireless ad-hoc networks. Proceedings of the 1st ACM international symposium on Mobile ad hoc networking and computing, pages:99-106, 2000.

[52] M. Bhardwaj, T. Garnett, and A.P. Chandrakasan, Upper bounds on the lifetime of sensor networks, IEEE International Conference on Communications, Pages:785-790 vol.3, 2001.

[53] D.M. Blough and P. Santi, Investigating upper bounds on network lifetime extension for cellbased energy conservation techniques in stationary ad hoc networks, in Proceedings of the Eighth ACM Mobicom, Pages:183-192, 2002.