CDAR: Contour Detection Aggregation and Routing in Sensor Networks

A Thesis Submitted to the

College of Graduate Studies and Research

in Partial Fulfillment of the Requirements

for the degree of Master of Science

in the Department of Computer Science

University of Saskatchewan

Saskatoon

By

Venkat Aveen Reddy Pulimi

Permission To Use

In presenting this thesis in partial fulfillment of the requirements for a Post-graduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made f any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science
176 Thorvaldson Building
110 Science Place
University of Saskatchewan
Saskatoon, Saskatchewan, Canada
S7N 5C9

Abstract

Wireless sensor networks offer the advantages of low cost, flexible measurement of phenomenon in a wide variety of applications, and easy deployment. Since sensor nodes are typically battery powered, energy efficiency is an important objective in designing sensor network algorithms. These algorithms are often application-specific, owing to the need to carefully optimize energy usage, and since deployments usually support a single or very few applications.

This thesis concerns applications in which the sensors monitor a continuous scalar field, such as temperature, and addresses the problem of determining the location of a contour line in this scalar field, in response to a query, and communicating this information to a designated sink node. An energy-efficient solution to this problem is proposed and evaluated. This solution includes new contour detection and query propagation algorithms, in-network-processing algorithms, and routing algorithms. Only a small fraction of network nodes may be adjacent to the desired contour line, and the contour detection and query propagation algorithms attempt to minimize processing and communication by the other network nodes. The in-network processing algorithms reduce communication volume through suppression, compression and aggregation techniques. Finally, the routing algorithms attempt to route the contour information to the sink as efficiently as possible, while meshing with the other algorithms. Simulation results show that the proposed algorithms yield significant improvements in data and message volumes compared to baseline models, while maintaining the integrity of the contour representation.

Acknowledgements

I take this opportunity to specially acknowledge and extend my gratitude to the people who made the successful completion of this thesis possible.

First and foremost I want to express my sincere appreciation to my supervisors, Dr. Derek Eager and Dr. Kevin Stanley for their incredible support throughout my thesis. Dr. Eager encouraged me to work in Wireless Sensor Networks as there was plenty of scope for improvement in this area, which I found it to be fascinating in course of time. Dr. Eager and Dr. Stanley steered me in the right path by providing me the necessary guidance and sharing their valuable knowledge during the meetings and discussions. They have taught me different ways to approach a problem which in turn helped me to improve my critical thinking and reasoning skills. I also appreciate the amount of effort they put in correcting my thesis.

I gratefully acknowledge my committee members, Dr. Dwight Makaroff, Dr. Mark Keil and Dr. Eric Salt (external) for their invaluable feedback on my thesis.

I am grateful to the office and technical staff members in the computer science department for assisting me in many different ways. In particular, I want to thank our graduate secretary Jan Thompson for providing timely help whenever needed.

I wish to thank each and every friend of mine for being on my side during the good and bad times. I am also very much indebted to the University of Saskatchewan Indian Student Association for providing me assistance during my initial days at Saskatoon.

I want to express my heartfelt thanks to Walter Bergen and Luella Bergen for treating me as a part of their family. The love, care and support they provided me throughout my time at the University of Saskatchewan made me feel like home away from home.

Lastly, and most importantly, I wish to sincerely thank my parents, Dinakar Reddy Pulimi and Sailaja Pulimi and my sister, Asritha Srireddy for their unconditional support and love which has been my greatest strength all through my life. I dedicate this thesis to my loving parents and sister.

Contents

List of Tables

List of Figures

List of Acronyms

WSN – Wireless Sensor Network

CH – Cluster Head

CM – Cluster Member

GN – Gateway Node

F/SPR – Flooding-based Shortest Path Routing

F/ATR – Flooding-based Aggregation Tree-based Routing

I/SPR – Information-driven Shortest Path Routing

I/ATR – Information-driven Aggregation Tree-based Routing

I/CATR – Information-driven Contour/Aggregation Tree-based Routing

I/CSPR – Information-driven Contour/Shortest Path Routing

CHAPTER 1

INTRODUCTION

Recent years have witnessed tremendous growth in Wireless Sensor Network (WSN) research. Various applications that involve sensing, monitoring, tracking and detection of a phenomenon make use of the WSNs to complete their tasks. This thesis focuses on design and performance issues for WSN applications in which the sensors monitor a continuous scalar field, such as temperature, and in where the primary task is to determine contour line locations in this field. Given a query asking for location of a particular contour line (for example, 20 degree temperature), the objective of the algorithms in this thesis is to detect the contour in an efficient and reliable manner, perform in-network processing to remove the redundant contour data and route the contour data to the destination sink node using an energy-efficient path.

The remainder of this chapter is organized as follows. Section 1.1 gives an overview of WSNs and the wide range of applications that make use of them. Important considerations in WSN design are discussed in Section 1.2. Contour-based WSN applications are discussed in Section 1.3. Section 1.4 summarizes the contributions of the thesis to the design and performance study of algorithms for contour-based WSN applications. Section 1.5 lays out the structure of the remainder of the thesis.

1.1 Wireless Sensor Networks

WSN applications involve sensing, tracking, and monitoring external phenomena. A WSN is an ad hoc network consisting of a collection of sensor nodes that are deployed within some region of interest. Figure 1.1 shows the clustered WSN topology that is used in thesis simulation experiments. Each node in a cluster is assigned a role and the node's

functionality is based on this role. Cluster Head (CH), Cluster Member (CM) and Gateway Node (GN) are the standard roles assigned within clusters discussed in this thesis. The CH carries out control, coordination and cluster processing functions such as propagation of a query within the cluster and aggregating member data before forwarding the data to the destination. Each GN acts as a bridge between two different clusters for forwarding messages and data between the clusters. Each CM carries the messages and data between the GNs and the CH. Aggregated data at each CH is forwarded to the sink using an efficient path.

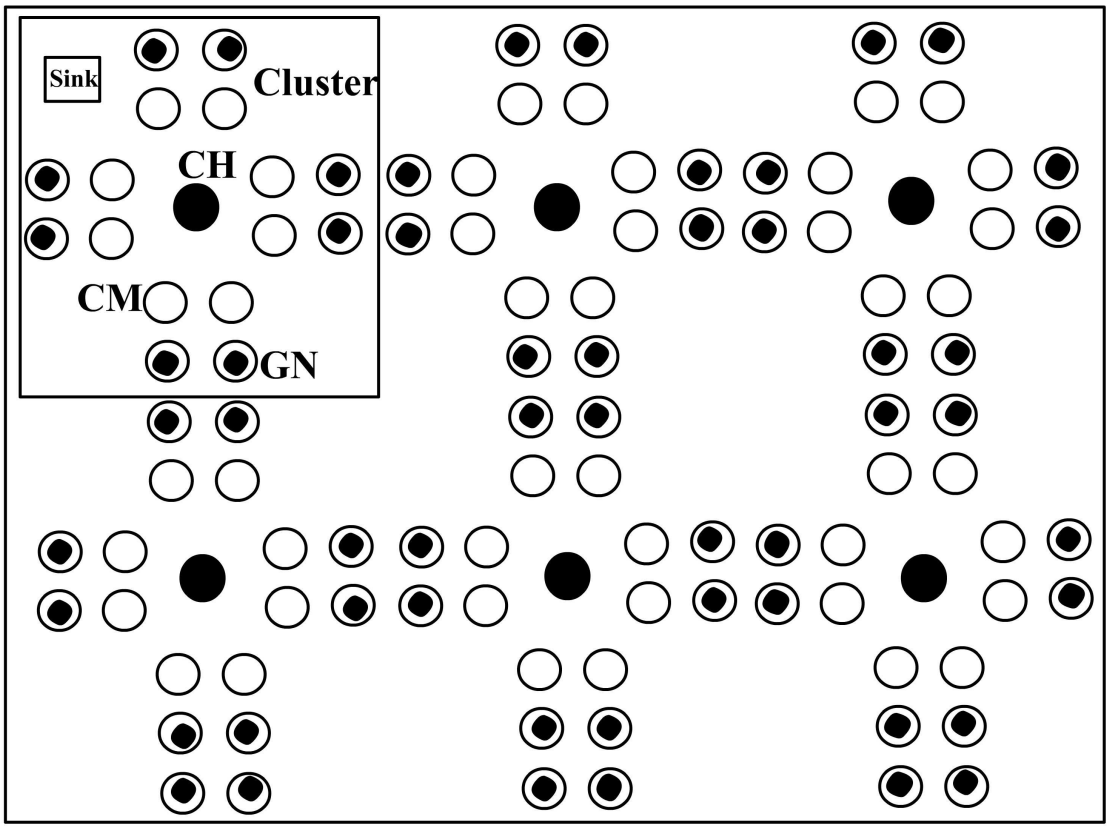

Figure 1.1: Clustered network topology

Each sensor node is comprised of a sensing unit, a processing unit, a transceiver unit and a power unit as shown in Figure 1.2. The sensing unit consists of sensors and analog to digital converters. The processing unit performs the information processing that allows the sensor node to collaborate with the other nodes to carry out the assigned sensing

tasks. The transceiver unit is responsible for the transmission and reception of data from the wireless medium. The power unit is used to hold the batteries to supply energy for the node to function. Since the nodes depend on batteries for their power supply, energy is a major constraint in these networks. Therefore it is desirable to provide an energy efficient solution to a WSN application that increases network longevity by reducing the amount of data transmission and reception by the nodes and yet provides accurate results.

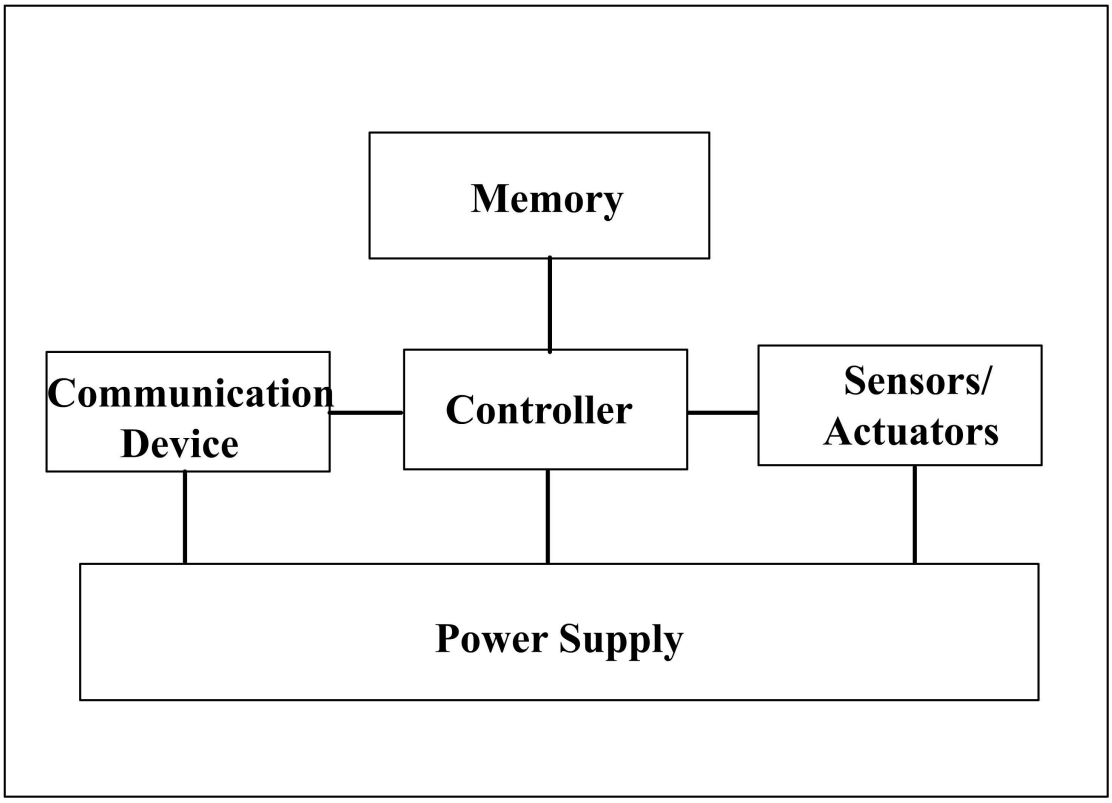

Figure 1.2: Sensor node block diagram [1]

WSNs have been gaining popularity due to their low maintenance cost, unattended operation, easy deployment and low hardware cost. Most of the data collection solutions in WSNs are application specific, because the potential energy savings of an application specific data collection solution outweigh the drawbacks of additional development. Also, deployments typically serve a single application only (rather than multiple

applications concurrently as does a computer system). WSNs are very broadly applicable, and can be used for detection, tracking, monitoring and controlling. These sensor applications can be typically classified as either event-driven or demand-driven. In an event-driven application, when a sensor node detects a particular event it informs the sink. For example, in case of a volcanic eruption, the nodes inform the sink on detecting the event. On the other hand, in demand-driven applications, sensors respond only if they receive a query from the sink. For example, consider a contour-based sensor application in which the location of a 10 degree temperature contour is needed. On receiving the query from the sink, nodes interact with each other to find if a 10 degree contour is detected in their vicinity. If so, the nodes that have detected the contour inform the sink. Spatio-temporal event monitoring, residual energy monitoring, faulty sensor detection and tracking targets such as animal, human, vehicle movements are some examples of different contour-based applications [2, 3]. A real-world coal mine surveillance application is used for detecting events using contours [4]. For the safety of the workers, the application detects two classes of events by deploying hundreds of sensors along the channels of the mine. One class of events is to detect gas, dust and water leakage. The other class of events monitors high and low oxygen density regions in the mine to ensure atmospheric quality.

1.2 Considerations in WSN Design

There are many challenges involved in designing algorithms for WSNs. It is not possible to find a single optimal design for all possible applications. WSNs are influenced by many factors, such as fault tolerance, scalability, resource constraints, topology control, transmission media and quality of service [5].

Fault tolerance: Nodes may fail due to loss of power, physical damage or environmental interference. The failure of a single sensor node should not compromise the overall task

of the WSN. Fault tolerance should be considered in schemes where node failures might hamper the completion of the required task, such as intrusion detection.

Scalability: Hundreds or thousands of nodes may be deployed in a single network. The performance of the protocols shouldn't deteriorate as network size increases.

Resource constraints: Power is used by the node for sensing, communication and data processing. Of these, data communication usually consumes more power than processing of the data locally and is proportional to the amount of data transmitted or received. The lifetime of a sensor network depends on the power resources of the nodes, which is constrained by the size of the nodes. Moreover, in remote deployments it is not feasible to replace the power sources present at the nodes. To avoid power depletion and increase the longevity of the network energy efficient algorithms should be employed.

Topology control: A node's transmission can be received by its neighbours. The number of nodes receiving this transmission is proportional to the node's transmission power. Greater power results in the node's transmission being received by neighbours that are further away and in some cases these received transmissions may be deemed unnecessary and discarded, wasting the sending node's power. Similarly, if the transmission power is too low, the node's transmissions may not be received by the nodes that should receive them. To avoid these problems, topology control schemes should be used to control each node's transmission power levels, implement network hierarchies and turn off unnecessary nodes.

Transmission media: In WSNs nodes usually communicate wirelessly using radio, infrared or optical signals that might encounter error prone channels and interference. To avoid these problems robust coding and modulation schemes should be used while transmitting the data.

Quality of Service: There is no fixed set of requirements that a sensor application must meet as the requirements vary from one application to another. In some applications, data reliability is a must for the application to work efficiently while for others it may be less critical. Similarly, some applications are tolerant to delay while delay may render the data useless in others.

These problems and challenges make WSNs an interesting research field, as there are ample opportunities for the development of efficient solutions. As explained before, solutions to a problem in WSNs are application specific. In this thesis a scalable, energy-efficient solution is described for contour-based WSN applications, by providing improvements in contour detection, query request propagation, in-network processing and query response routing.

1.3 Contour-based WSN Application

A contour-based application gives an overview of the phenomenon across a sensor field by constructing contour lines from the sensor readings. A contour line or isoline is a curve that connects points of similar value. Natural contours are generally continuous and smooth. Moreover, these contours are not uniformly spread throughout the network and pass close to only a subset of nodes. Contour line is a generic term used for any kind of phenomenon whose value can be described by a real number at different points in space and/or time. However, based on the monitored phenomenon, the term can be made specific. For example, isotherm is a line that connects points on a map with the same temperature.

For a node to detect the presence of a contour in its vicinity, it has to receive the sensed readings from the neighbouring nodes. On receiving the readings, the node can compare its sensed reading to that of the readings received from its neighbours. If these readings lie on either side of the contour value defining the contour line, then the presence of a

contour is detected by the node. For example, consider a query in which the sink is interested in a 10 degree temperature contour. If a node's sensed temperature is 9 degrees and that of a neighbour node is 11 degrees then a 10 degree contour exists between these nodes.

This thesis considers specifically contour-based applications in which queries for the current location of a particular contour line are issued by the sink node. Flooding is one method to propagate the query into the network and it results in the query being propagated through the entire network. Since a contour line is typically not present throughout the entire network, this approach can be inefficient. Random walk, gradient routing or contour trees can be used instead. Random walk and gradient routing schemes route the query greedily based on local information [30-34]. Contour tree schemes, on the other hand, preprocess the signal field and construct a contour tree based on the contour values enabling the query to be routed along the tree efficiently [3].

In a sensor field, spatial and temporal correlations in data exist. Forwarding the raw contour data without performing in-network processing results in unnecessary resource wastage. Moreover, the sink may not be interested in all of the received correlated data. In-network processing techniques such as data aggregation and data compression can be performed to remove the redundant data [6]. Data aggregation uses various aggregation functions such as MAX, MIN or AVG or application-level parameters to suppress the redundant data. Even after aggregation is performed the actual data that is being transmitted can be further compressed using various encoding techniques. These processing schemes can be applied locally at the node level or globally in a distributed manner while the contour data is propagated to the sink. Forwarding of the processed contour data to the destination is usually done along the reverse path of the query, or the shortest path using an aggregation tree or independently. An aggregation tree is a minimum spanning tree with the sink or destination as its root, on which data aggregation takes place while the data is being propagated to the sink.

Existing research mainly focuses only on particular aspects like contour detection, query propagation, in-network processing or data routing to improve the efficiency of a contour-based WSN application [2-4, 31-41]. There are a number of problems with the current schemes. In some of the current approaches, efficient distributed in-network processing and routing of the contour data to the destination using an aggregation tree can be done only when the query is flooded in the network. However, the gain obtained due to efficient in-network processing and data routing may be small compared to the cost of flooding the network to propagate the query. To avoid flooding some approaches use an efficient algorithm to propagate the query such as random walk, gradient routing or contour trees to detect the contour and then route the data in the reverse path of the query or shortest path after aggregating along the contour. These techniques do not guarantee that the data is routed to the destination in an efficient manner. Reverse paths to the destination are not always the shortest paths and any hop in the reverse path that is not on the shortest path to the sink may result in resource wastage. Similarly, aggregating the contour data along the contour and routing the overall data to the sink in the shortest path is also expensive. The cost of routing may exceed the query flooding cost due to the large data payload if the route is suboptimal. In this thesis, all these problems are addressed in detail and an overall efficient end-to-end solution for contour-based WSN applications is proposed.

1.4 Thesis Contributions

This thesis focuses on providing a novel overall efficient solution for contour-based sensor applications. The proposed solution includes a method of propagating and processing a query so as to find a point on the requested contour, an efficient and reliable manner of propagating the query along the contour, efficient in-network processing so as to remove redundant contour data, and data routing along an efficient path. Moreover, the proposed approach provides the flexibility to incorporate other distributed in-network processing schemes. The main contributions of this thesis are:

- Methods for query routing and processing that find a point on the requested contour and then propagate the query along the contour in a reliable and efficient manner.
- In-network processing techniques using both aggregation and compression algorithms to remove and reduce the redundant data while propagating the data to the sink.
- Data routing algorithms that work well with the proposed contour detection, query propagation, and in-network processing schemes and route the contour data to the sink in an efficient manner.

1.5 Thesis Organization

The remainder of the thesis is organized as follows. Chapter 2 gives an overview of various basic WSN techniques which are used by the proposed algorithms. Chapter 3 describes related research on contour-based WSN applications. Chapter 4 describes the design of the algorithms in detail. Chapter 5 discusses the simulation methodology for performance evaluation. Chapter 6 presents results from the simulation experiments that are carried out. Chapter 7 summarizes the thesis and outlines possible areas for future work.

CHAPTER 2

BACKGROUND

WSNs provide the capability of monitoring a particular phenomenon without significant human intervention. The flexibility they provide creates various problems and challenges, as explained in the previous chapter. Moreover, the solutions to these challenges are mostly application specific. This chapter presents background concerning WSNs that is relevant to the work in this thesis. Section 2.1 gives an overview of different network topologies in WSNs. Interest propagation by the sink using different queries is explained in section 2.2. Section 2.3 explains different in-network processing techniques to remove redundant data. Section 2.4 describes different routing techniques to route the information in WSNs.

2.1 Network Topology

In WSNs, issues like signal interference, multiple transmission routes to the destination and reconstruction of routes in case of node failure are known. These problems can be overcome to a certain extent by topology control schemes. A topology scheme provides reliability, high throughput, connectivity, energy efficiency and potentially mobility by controlling each node's transmission power level or through careful selection of those nodes within transmission range that will be used for packet forwarding (i.e., will be neighbours in the network routing topology). Most commonly, topologies are formed by either controlling the transmission power or by imposing a hierarchy onto the network.

2.1.1 Network Hierarchies

In hierarchical topologies, the emphasis is on selecting a set of nodes which are assigned special coordination, control, and for routing responsibilities compared to the rest of the nodes present in the network. This can be achieved either by clustering or backbone-based techniques.

2.1.1.1 Backbone-Based

A subset of nodes which form the backbone are selected such that the each other node is connected to at least one of these backbone nodes and the backbone is connected to the sink. This is an example of the Connected Dominating Set (CDS) problem. The backbone is created by constructing trees, connecting independent sets and/or by pruning techniques. Trees can be constructed using centralized or distributed approaches. Prim's algorithm [7] is used for constructing a minimum spanning tree in a centralized manner. On the other hand, the A3 protocol proposed by Wightman and Labrador [8] constructs a tree in a distributed manner using node energy and distance information. In the second approach, independent sets are created and then connected to form the backbone. An independent set consists of nodes that don't have any edges between them. These independent sets might not be connected, so in the later stage a minimum set of nodes are selected to connect these sets. The Energy Efficient Connected Dominating Set (EECDS) algorithm proposed by Zeng *et al.* [9] uses this approach. The final approach uses pruning techniques to reduce unnecessary nodes selected in the backbone and yet maintain the connectivity. It can be used with the other techniques mentioned to get the initial set of nodes and later prune them accordingly. The Connected Dominating Set under Rule K (CDS-Rule-K) algorithm proposed by Wu and Dai [10], and Wu and Li [11] makes use of this approach.

2.1.1.2 Clustering

The network is partitioned can be partitioned into several clusters. The nodes within each cluster may be assigned different roles. For example, in the topology assumed in this thesis each cluster consists of Gateway Nodes (GNs), Cluster Members (CMs) and a Cluster Head (CH), which may be dynamically determined or statically configured. The members perform the sensing operations and transmit the sensed data to the CH; the CH may perform data aggregation before routing the data via the neighbouring clusters to the sink. Routing to the neighbouring clusters is done by a GN. Use of a hierarchical topology can save node energy and prolong the network lifetime because nodes do not transmit data individually to the sink. However, if no aggregation occurs, the additional overhead can reduce efficiency. Gerald and Tsai [12] have proposed a clustering approach based on the highest degree heuristic. In this approach, a node is selected as a CH if it has the highest number of neighbours. Baker and Nephritides [13] have proposed a clustering approach that uses lowest node ID heuristic in CH decision making. Hein Zelman *et al.* [14] have proposed a clustering algorithm called Low Energy Adaptive Clustering Hierarchy (LEACH). In LEACH a sensor node chooses a random number between 0 and 1. If the number is less than a threshold value then the node becomes a CH for the current round. Chattered *et al.* [15] have introduced a Weighted Clustering Algorithm (WCA) which heuristically combines several attributes such as battery power, node degree, transmission power, and node mobility into a single weight parameter, which is used in the election process for choosing a CH.

2.1.1.3 Controlling the Transmission Power

The objective of transmission power control is to build a reduced topology while maintaining an overall connected network. Location, direction and/or neighbour count information can be used to set the transmission range in a distributed manner. The Local Minimum Spanning Tree (LMST) protocol proposed by Li *et al.* [16] is a distributed location-based topology control scheme in which each node creates a Euclidean Minimum Spanning Tree (EMST) from the location information of the neighbouring

nodes. Later, it sets a transmission range that allows it to reach its farthest neighbour in the EMST. Directional information can also be used for controlling the range. The direction of the incoming angle of the signal can be detected if the node has a directional antenna and the distance using various techniques like the Received Signal Strength Indicator or Time of Arrival. The Yao Graph algorithm proposed by Yao [17] is a directional-based topology control scheme which woks in two phases. In the first phase, the original network is partitioned into a sub-network based on the MST and in the final phase it is pruned. The final technique is based on the node's neighbours. The goal is to connect a node with a minimum number of neighbours and minimum power and yet maintain the network connectivity. The K-Neighbour protocol proposed by Blough *et al.* [18] ensures that each node is connected by the number of neighbours specified by the parameter K.

In CDAR, algorithms are built on a statically configured two-hop cluster-based hierarchical topology control scheme which ensures better network connectivity. In a two-hop cluster the maximum hop distance between the CH and any node on the cluster boundary is utmost two-hops. However, the proposed algorithms can be extended to multi-hop clusters also. As explained in the clustering section, each of the nodes is assigned certain roles to perform within a cluster and the assignment of these roles is done through an election process. However, in the algorithms we assume the roles of the nodes within the cluster are also pre-configured. Moreover, maintenance or fault tolerance of the clusters is not addressed, as it is outside the scope of the thesis. However, any geographically aware clustering scheme could be integrated with the proposed algorithms.

2.2 Queries

In contour-based applications, preprogramming the nodes with some specific contour values makes it difficult for the user to change them to different values at a later point of

time. To avoid this problem, the sink can specify these values in a query and propagate the query into the network. For contour applications, apart from specifying the contour value in the query, other parameters for performing in-network processing may also be specified. Most of the sensor nodes store the data received from their neighbours in the form of records in a table. Parsing of these records can be done using queries in Structured Query Language (SQL) or using application specific programming languages. A query template in the SQL language was suggested by Yao and Gherkin [19] :

SELECT {aggregates (attributes)}
FROM {Sensor data S}
WHERE {predicate}
GROUP BY {attributes}
HAVING {predicate}
DURATION time interval
EVERY time span e

The SELECT clause is used to extract data and represent it in a user friendly manner as specified by the attributes and aggregates. These aggregates are functions like MAX, MIN and AVG which specify the form of performing in-network processing. The FROM clause is always followed by the SELECT clause and specifies the table from which the data is to be retrieved. The WHERE clause is optional, when specified it always follows a FROM clause and is used to filter the data from the table based on the predicate. More than one condition can be specified in the WHERE clause using the logical expressions like AND, OR, LESS THAN or GREATER THAN. The GROUP BY clause is used together by the aggregate functions to group the retrieved data. The HAVING clause can be used with a SELECT clause to specify a search condition for a group or aggregate. It behaves like a WHERE clause, but is applicable to groups. The DURATION clause specifies the life time of the query beyond which the query expires. The EVERY clause specifies the interval after which the node should sense the external phenomenon, if the query is periodic.

Different Query Types

Based on the applications, different queries can be disseminated into the sensor network. Different query types and examples from Madden *et al.* [20] are explained in detail below. Periodic queries are executed at regular intervals for a specific duration. For example, the following query indicates the node to sense the temperature every 5 seconds for the next 100 seconds and report the sensed data to the sink.

```
SELECT Temperature
FROM SensorTable
DURATION (now, now +100)
EVERY 5 seconds
```

These types of queries are useful in environmental monitoring applications. These applications require the status of the monitoring phenomenon to be reported at regular intervals, so that the decisions can be made accordingly. The STOP ON EVENT clause can be used to stop a periodic query when a specified event is triggered based on satisfied condition.

Event-based queries are executed only when the predefined conditions are satisfied. These queries are useful in tracking or detection applications. For example, the following query indicates the node to sense the surrounding temperature only when the node's pressure sensor detects a pressure of greater than 30 Pa.

```
ON EVENT Pressure > 30
SELECT Temperature
 FROM SensorTable
```

Life-time based queries are similar to the periodic queries, but the sampling rate is based on the remaining power available, so that the condition specified in the LIFETIME clause is achieved. For example, the following query indicates the node to sense the temperature

for a month and report the sensed data to the sink. The node changes its sensing period based on the power available, so that it can sense and report the data for a month.

```
SELECT Temperature
FROM SensorTable
LIFETIME 1 month
```

Exploratory queries indicate the nodes to monitor the external phenomenon only once. The ONCE clause is used to achieve this functionality. For example, the following query indicates the network to sense the temperature once and report the data to the sink.

```
SELECT Temperature
FROM SensorTable
ONCE
```

Actuation queries are used to turn on some components in the sensor if the condition in the query is met. For example, the following query indicates the node to sense the temperature every 5 seconds for the next 100 seconds and report the sensed data to the sink only if the temperature is greater than 40 degrees. In addition to reporting the data, the node turns on the fan.

```
SELECT Temperature
FROM SensorTable
WHERE Temperature > 40 degree
OUTPUT ACTION turnOn (fan)
DURATION (now, now +100)
EVERY 5 seconds
```

As explained above, queries are a means of encapsulating the interest of the sink in a message and propagating it through the network. In the algorithms, an exploratory query is used to propagate the query from the sink pertaining to the contour through the network. However, the proposed algorithms are extensible to periodic or lifetime queries. Once the exploratory routing algorithms are established, periodic and lifetime queries can

be implemented with existing techniques. Moreover, an application specific programming language can be used to encode the query.

2.3 In-network Processing

Forwarding sensor data to the sink is costly, as it consumes significant energy per byte sent. Performing in-network processing at intermediate nodes while transmitting the sensor data to the destination increases the network lifetime by reducing the total number of bytes sent. Aggregation and compression are the most common in-network processing techniques. Aggregation is a primary concern in the proposed algorithms. In this section, general aggregation approaches are presented and specific contour aggregation approaches are presented in section 3.2.

Aggregation

Aggregation is a technique used to suppress or combine individual sensor data at intermediate nodes while transmitting the data to the destination. It can be performed by applying general aggregation functions on the sensor data or by applying application specific parameters set by the sink. The sensor data packet header overhead can also be reduced by aggregating different sensor data packets into a single packet, and in the process multiple packet headers can be removed. Most of the hierarchical topology schemes discussed in section 2.1.1.2 are good candidates for performing data aggregation. Aggregation functions are generally specified in the queries by a controlling node or sink in dense hierarchical networks where the data is often correlated. For example, readings taken of a temperature field are often spatially correlated. MIN, MAX, AVG, COUNT and SUM are some of the basic aggregate functions that can be used to perform aggregation as described by Madden *et al.* [20] If an intermediate node receives two partial state records <a> and <b> from different nodes, an aggregation function computes a new state record, <c> = f (<a>, <b>). A partial state record is a tuple

exchanged between the nodes. He *et al.* [21] have proposed an Adaptive Application-Independent Data Aggregation (AIDA) approach in which payloads are concatenated resulting in header transmission savings.

Data Compression

At times it is not possible to reduce the size of the sensor data using aggregation. For example, consider content-sensitive data like fixed-width histograms as explained in Madden *et al.* [20] which can't be reduced using normal aggregation without losing information. As explained in the previous section, aggregation reduces the sensor data at intermediate nodes. However, the content-sensitive data and aggregated data can be compressed at the intermediate nodes using standard lossless data compression algorithms before transmitting, if correlations are present in the sensor field. S. S. Pradhan *et al.* [22] have proposed a distributed source coding framework for efficient compression in a WSN. Hellerstein *et al.* [6] have proposed a compression scheme to encode wavelet histograms in WSNs.

In CDAR, suppression of redundant sensor data is performed at the cluster-level using contour application-specific parameters specified by the sink. At the cluster-level, data suppression is performed at the GNs, CMs and CH before transmitting the data to the destination. Finally, the response data packet headers are suppressed by aggregating different response data packets into a single packet at the intermediate nodes as in [21].

2.4 Routing

In WSNs, nodes that sense the data may not transmit the sensed data directly to the destination, due to network size and transmission energy constraints. These source nodes rely on the intermediate nodes to route the sensed data to the destination. Routing tables

are used to forward the sensed data to an appropriate neighbour before it can reach the destination. Apart from routing the data to the destination, construction and maintenance of the routing tables are the primary responsibilities of a routing protocol. Data transmission and routing table construction and maintenance between the nodes is done by unicast, broadcast or multicast. Unicast transmits the data from the source node only to the destination node. Broadcast transmits the data to all the nodes in the vicinity of the source node. Multicast transmits the data to a subset of nodes in the vicinity of the source node.

Network topology and application dependence play important roles in routing. Depending on the network structure routing protocols can be divided in to flat routing, hierarchical routing and location-based routing schemes. Similarly, based on the protocol operation, the routing protocols can be multi-path based, query-based and negotiation-based. All of these protocols, irrespective of the network topology or protocol function, fall in two main categories: reactive and proactive routing protocols [23]. Reactive protocols find routes to the destination only when needed, whereas proactive protocols find the routes beforehand. Proactive protocols have a large signaling overhead compared to reactive protocols because periodic and event based route updates are required to update routing tables. However, proactive protocols have a low latency as the routes are already known to the destinations. The main goal of any routing protocol is to prolong the network life time, reduce the signaling overhead and therefore reduce the energy consumption. Our algorithms require proactive protocols to be effective.

2.4.1 Network Topology

Routing protocols are classified into flat routing, hierarchical routing and location-based routing. In a flat routing technique, all the nodes perform a similar functionality. Destination Sequenced Distance Vector (DSDV) routing proposed by Perkins and Bhagwat [24] is a flat proactive routing approach based on modifications to the

Bellman-Ford algorithm. Dynamic Source Routing (DSR) proposed by Johnson *et al.* [25] is a reactive routing approach. Route discovery is performed to the destination on demand if a route doesn't exist in the node's table while transmitting the data. In hierarchical routing schemes, nodes are assigned different tasks to perform. Low Energy Adaptive Clustering Hierarchy (LEACH) proposed by Heinzelman *et al.* [14], implements hierarchical routing. Geographic routing schemes address the nodes based on their position. Takagi and Kleinrock [26] have proposed the Most Forward within R (MFR) algorithm in which a source node forwards a packet to the neighbouring node within its transmission range and whose position is closest to the destination.

2.4.2 Application Dependence

Some routing protocols are specific to WSN applications. If an application demands fault tolerance in the network then multiple paths should be established between the source and destination in the network. Furthermore, establishment of multiple paths to the destination helps maintain uniform energy consumption along different network paths. Chang and Tassiulas [27] have proposed an approach in which the nodes use the path that contains the largest energy. If the energy of this path falls below any of the existing multiple paths than the next highest energy path is selected to route the data. In some applications, the destination may be interested in different kinds of data from the network. If a source node senses the external phenomenon and finds that it has the data that is required by the destination, then the node routes the data along the route through which it has received the query. Intanagonwiwat *et al.* [28] have proposed a data centric paradigm called Directed Diffusion. An interest is disseminated through the network and gradients are setup matching the interest. Data flows through the gradients along multiple paths to the initiators of the interest. In-network aggregation is performed along these paths eliminating data redundancy and prolonging the network life. In certain applications, negotiations between nodes are performed regarding various resource attributes in order to increase the energy efficiency of the network. Kulik *et al.* [29] have proposed Sensor Protocols for Information via Negotiation (SPIN) which disseminates

information among sensors in an energy efficient manner through data negotiation. Nodes that have new data disseminate by advertising about the data to their neighbours. Neighbours may request for the data or ignore the advertisement. On receiving requests, the advertised node forwards the data to the nodes that have requested for the data. In this manner, any new data gets propagated to all the nodes present in the network. User can query any node in the network to get the required information.

In CDAR, routing tables are statically configured for a two-hop cluster-based hierarchical topology. Nodes contain routes to the sink, their own CH and the neighbouring CHs in the node's vicinity. Route maintenance is outside the scope of the thesis. A hierarchical cluster-based routing scheme is used for routing the query from the sink to the phenomenon. For routing the sensed data from the network back to the sink a novel combination of hierarchical cluster-based routing and tree-based routing schemes are used.

CHAPTER 3

RELATED WORK

The thesis focuses on providing an overall efficient solution for query-based contour sensing WSN applications. Most contour-based applications perform contour detection, query propagation, in-network processing and data routing. Though these phases are common to many WSN applications there are significant improvements that can be made to these phases for a contour application. These improvements are based on the simple fact that contours are not spread uniformly throughout the field and are usually continuous and smooth. Moreover, exploiting the spatial and temporal correlations between the nodes along the contour helps to improve the performance. There are three major areas of study in contour-based WSN applications. First, contour detection and query propagation techniques used for detecting the contour and propagating the query are studied; second, in-network processing schemes used for removing redundant contour data are studied; third, routing of the sensed contour data to the destination is presented.

3.1 Contour Detection and Query Propagation Mechanisms

Contours are generally not spread uniformly throughout the network. Exploiting this fact and propagating the query only along the nodes that have detected the phenomenon avoids unnecessary query propagation overhead. A number of query propagation mechanisms have been proposed in the literature to efficiently route the query for different WSN applications, not limited to contour-based applications.

Intanagonwiwat *et al.* [28] have proposed a publish-subscribe and application-aware paradigm which uses diffusion to achieve energy savings by setting up gradients between the source and the sink. This enables the sink to propagate the query to certain parts of

the network where the interest may be present. This approach is suitable for persistent queries which are used to monitor a phenomenon that doesn't change over a period of time. In certain environmental monitoring applications, the node that has detected an event floods the network with the event establishing energy efficient gradients towards the event. Event flooding is advantageous if the number of events generated by the network is small and the scope of these events is small. Rumor routing proposed by Braginsky and Estrin [30] provides a cut off value under which rumor routing provides an energy-efficient solution compared to flooding. A node that has detected an event establishes the event paths by forwarding the information about the event to the nodes based on a specified threshold. If the number of events increases, there is an overhead in maintaining the events at each node and the propagation of the event information to the neighbours also increases.

Sadagopan *et al.* [31] proposed a technique for querying sensor networks called ACtive QUery forwarding In sensoR nEtworks (ACQUIRE). In this approach, the node forwards the query from one node to another using random walk based on the event information until the query has been fulfilled. The algorithm uses a look-ahead parameter which specifies the neighbouring nodes from which the node can request the updates to resolve the query. Both the query and response phases are performed in a single step. As the query is propagated, the partial results generated by the nodes are aggregated. Finally on resolving the query, the data is forwarded to the destination using the shortest path or the reverse path. Chu *et al.* [32] have proposed information-driven sensor querying (IDSQ) and constrained anisotropic diffusion routing (CADR) routing techniques which are based on the directed diffusion. These techniques use information gain and communication cost to perform energy efficient routing.

Liu *et al.* [33] proposed an information-directed multiple-step look-ahead approach to route the query to the event. It generalizes the CADR approach. This approach allows a node to search a path with maximum aggregation from the available paths within the look-ahead range. Selecting a proper look-ahead parameter value allows the nodes to

avoid sensor holes while query routing. A sensor hole occurs in a WSN when neighbouring nodes fail, and is defined as the region containing these failed nodes. Faruque and Helmy [34] have proposed a distributed scheme called RoUting on finGerprint Gradients in sEnsor Networks (RUGGED) which uses the natural gradient in routing the query without much overhead. It uses multiple paths to find a route to the event and it controls these multiple paths using a probabilistic function. It uses two modes of operation: flat region and gradient region. In the flat mode, all the neighbouring nodes are queried for the gradient information. If a gradient is detected, then the node switches to the gradient mode and uses a greedy approach to route the query. If there is not sufficient information gain using the gradient mode, then probabilistic forwarding is performed. Sarkar *et al.* [35] have proposed an approach that guarantees delivery of the query for a static contour field. This is done by pre-processing the field and constructing a contour tree and routing the query in a gradient-based manner along the tree. A contour tree is a tree on all the critical points of the signal field and captures all the contours. Zhu *et al.* [3] have proposed a light-weight distributed algorithm for contour tracking and repair of broken contours locally as they deform. It is feasible for signal fields which don't deform rapidly and is suitable for periodic queries.

3.2 Contour In-network Processing Schemes

Nodes that have the sensed contour data perform in-network processing before routing the data to the sink. In-network processing helps in removing data that the sink might not require. Processing of the data can be done at the node level before it is transmitted the sink or at the intermediate nodes while the data is being transmitted to the sink or at both.

Hellerstein *et al.* [6] have proposed an approach for constructing a contour map within the network through identification of contiguous regions (termed "isobars" in this work) in which the sensors have approximately the same value. A comparison is performed between the naive, in-network and lossy approaches. The naive approach constructs the

contour map outside the network; in-network approach constructs the contour map within the network using aggregate functions; and the lossy approach restricts the vertices used to define the bounding polygon of each isobar by a parameter. Of all these approaches, the lossy approach reduces the data significantly.

Zhao *et al.* [36] proposed residual energy scan (eScan) which uses in-network aggregation to indicate the remaining energy levels of sensor nodes in the network. It uses a polygon-based aggregation technique to aggregate the eScan reports generated by nodes while transmitting the reports to the destination using an aggregation tree. Buragohain *et al.* [37] proposed an Adaptive-Group-Merge polygon-based aggregation technique that constructs a k-vertex polygon for a given parameter k.

Xue *et al.* [4] have proposed an in-network map construction using a partial map aggregation technique, hop-by-hop in a bottom up manner. A partial map generated by a node consists of disjoint contour regions. On receiving partial maps from its children, the node includes its own partial map with the received ones and tries to merge the adjacent contour regions and generates a final partial map which is transmitted to its parent. In order to reduce the transmission size of the final partial map, compression and packet snooping is performed at the node. Incremental map updates also reduce the size of these transmitted maps considerably. In all these schemes, the overhead due to message transmission is high because all the nodes transmit the messages. Moreover, the polygon-based aggregation schemes need to encode the location information of the nodes which is costly. Sometimes, aggregation of spatially correlated readings of adjacent nodes cannot immediately be performed using an aggregation tree, unless they have a common parent, resulting in a significant overhead. Every node in the network must be powerful enough to process the sensed data as it is transmitted to the sink.

To avoid unnecessary transmissions by all the nodes, Solis and Obraczka [38] have proposed an approach called isoline aggregation which creates a contour map at the sink

from the reports generated by the isoline nodes rather than aggregating readings from all the nodes in the network. These isolines are detected from the local neighbour information using the neighbour-to-neighbour protocol. A node is said to be an isoline node if its sensing value and its neighbour's value are on either side of the isolevel specified in the query. Only nodes that detect the isoline report to the sink. These reports consist of the node ID and the neighbour node ID which help in detecting the contour. The sink reconstructs the contour map based on the node information received. Liu and Li [39] have proposed a similar approach called the isomap approach to create a contour map at the sink. Isoline nodes perform linear regression on the values received from the neighbours and find a gradient direction which is used by the sink for contour map construction. Li and Liu [40] have used angular separation and distance separation to suppress the responses spatially at the intermediate nodes.

Meng *et al.* [2] have proposed distributed spatial and temporal data suppression, multi-hop local suppression and contour construction at the sink using interpolation and smoothing. Contour readings generated are generally spatially and temporally correlated resulting in redundant transmissions. A node suppresses its transmission if the difference between the average reading values of the neighbouring nodes is within the threshold value. On the other hand, temporal suppression controls the node's transmission rate. This approach reduces the data overhead by suppression. However, the accuracy of the contour map approximation at the sink is dependent on the amount of suppression performed in the network.

Singh *et al.* [41] have proposed a technique for constructing the contour map in a distributed manner using divide and conquer approach in a cluster-based topology. The CHs on receiving the data from their members aggregates the data locally. In the first round, within each block of 2x2 CHs, one CH is elected as a leader and merges the data received from the other three CHs. In the next round, the block leaders organize themselves into 2x2 blocks and a leader is elected among them and merging of data is

performed. In this manner, merging of the data is performed recursively constructing the contour map.

Yoon and Shahabi [42] proposed Clustered Aggregation (CAG) algorithm which exploits the spatial correlations among the members in a cluster and suppresses the locally within a cluster before sending just only one value per cluster up the aggregation tree. In-network aggregation is performed along the aggregation tree using various aggregation functions at the intermediate nodes while the data is being propagated to the sink. Pattem *et al.* [43] have shown that a static clustering scheme provides an optimal performance for a range of spatial correlations compared to suppression of spatial correlated data while routing. The performance of the latter is dependent on the level of correlations present in the data.

3.3 Contour Data Routing Techniques

Nodes that have the sensed data must forward the data to the destination over the shortest possible path in order to avoid unnecessary energy loss. Intermediate nodes may or may not perform in-network processing on the raw data while the data is being transmitted to the sink. Performing in-network processing at the intermediate nodes helps by removing unnecessary data that the sink might not require and hence save the transmission cost. Most of the applications that use query propagation schemes like random walk, contour trees or gradient routing to propagate the query in an efficient manner either route the contour data back to the destination using the reverse or shortest paths [31-35].

3.4 Contour Applications Research Problems

Contour detection using random walk or gradient routing may result in the contour query getting trapped at the saddle point, local minima or local maxima because they use

information gain and communication cost to route the query. Moreover, these approaches are not feasible for finding all the contours present in the network. Contour trees avoid this problem of query getting stalled because of saddle point, local maxima or minima by pre-processing the field before the query is propagated. However, for a dynamic contour field where the field changes constantly, the contour tree approach is not feasible, as the overhead for tree construction and reconstruction is high. In contour-based WSN applications, nodes need to know their location and indicate the location to the sink for it to construct the contour map. Based on this requirement, the proposed contour detection algorithms use the propagation pattern and look-ahead range in detecting a contour. The propagation pattern in the query contains details of how and where the query needs to be routed. This can prevent the query from being stalled because of saddle point, local maxima or minima if a blind or geometric route is specified. Once the query is propagated according to the pattern, the look-ahead parameter is used to search for a contour in the vicinity of the pattern. This flexibility is difficult to achieve when random walk or gradient routing is employed because the query propagation path is not known to the sink. If multiple contours must be found or for large networks, multiple propagation patterns can be included in the query for the nodes to route the query to different locations based on the specified patterns.

On finding a node near the contour, the current schemes propagate the query randomly or greedily based on information gain. To avoid this effect clustering is used, helping that the CH can make a reliable decision based on all the received member readings. Our work is similar to IDSQ routing proposed by Chu *et al.* [32] which uses cluster-based information gain to propagate the query. However, they do not explain how the query propagation takes place between clusters. In CDAR, the propagation technique to propagate the query efficiently and reliably along the contour using clusters is explicit. Moreover, contour tracking is done accurately using a look-ahead range rather than using a greedy approach. The min-hop routing algorithm proposed by Liu *et al.* [33] also uses a look-ahead parameter to avoid the query getting trapped in saddle points, local maxima or minima. However, their approach increases the neighbourhood size resulting in

unnecessary communication overhead. In CDAR, the look-ahead range adjusts automatically depending on whether the contour is detected, reducing the overhead.

Global distributed in-network processing schemes like polygon-based aggregation require the nodes to encode their location information in order to perform aggregation which is costly. Sometimes, aggregation of spatially correlated readings of adjacent nodes cannot be performed using an aggregation tree, because the common parent occurs many hops up the tree. To avoid these problems, a localized cluster-based in-network processing technique is proposed which uses data aggregation and compression for removing redundant contour data in an efficient manner. Pattem *et al.* [43] have shown that a clustered topology with optimal cluster size would perform well for a wide range of spatial correlations. Yoon and Shahabi [42] proposed Clustered Aggregation (CAG) which uses a similar outline to perform aggregation, but the aggregation proposed in CDAR is completely different. CAG is a lossy approach; only the CHs perform the in-network processing using aggregation functions and each cluster reports a single value. Moreover, the clusters are created each time based on the threshold in the propagated query. In CDAR, clusters don't change with the query and the nodes in the cluster that detect the contour perform different levels of in-network processing internally within the cluster while propagating the data to the CH. Further processing of the received member data is done at the CH before the data is routed to the destination. In addition to the in-network data processing, the number of intra-cluster control message transmissions is also reduced.

Meng *et al.* [2] perform suppression of spatially correlated data using a constant reading value set by the sink. However, the accuracy of the contour map approximation at the sink is completely dependent on the amount of suppression performed in the network. The proposed suppression scheme is based on a minimum contour threshold distance parameter set by the sink which indicates the minimum distance between any two points on the contour that the sink is interested in. Any contour points between these minimum distance contour points on the contour are suppressed. This gives the user the flexibility

to control the amount of suppression by varying the threshold parameter and yet construct a contour map accurately. Furthermore, efficient compression techniques are provided to reduce the size of the encoded payload data. Li and Liu [40] use angular separation and distance separation to spatially suppress the reports which is similar to the proposed in-network spatial suppression schemes in this thesis. However, the approach proposed by them requires transmission of node location to perform suppression at the intermediate nodes which is expensive. If the distance separation is smaller then a lot of responses are not suppressed even if they are close, resulting in a greater overhead because of the location encoded in these responses. Moreover, there is a chance that these responses can take different paths up the aggregation tree and might get aggregated at the higher nodes, wasting resources. By taking into account smoothness and continuity of the contours, local in-network spatial suppression schemes are proposed which avoid transmission of location information in the responses, reducing the unnecessary overhead.

Most of the applications that use efficient query propagation schemes either route the contour data back to the destination using the reverse path or the shortest path. Aggregating and routing the data back to the destination in the reverse path after the query is resolved may result in significant wastage because of the potentially large payloads in contour applications, as the reverse path is not always the shortest path. Similarly, aggregating the data along the contour and finally propagating the aggregated data to the destination in the shortest path is also costly. Moreover, routing each node's contour information to the sink individually is expensive if the network is not clustered. Additional overhead due to the individual packet headers for every data point is incurred. In these approaches, all the savings obtained by propagating the query efficiently are compensated by inefficient data routing techniques. The novel data routing algorithms proposed in CDAR help in routing the data in an energy-efficient manner to the destination and while allowing efficient query propagation and in-network processing.

CHAPTER 4

PROTOCOL FRAMEWORK

This chapter presents the design of the energy efficient end-to-end CDAR protocol framework for contour detection, query propagation, in-network data processing and routing of the contour data to the sink. All the proposed algorithms are designed specifically for contour-based WSN applications by taking the natural properties of the contours like smoothness, continuity and correlations into consideration. Apart from the proposed algorithms, some well known algorithms are also presented in this chapter because they are used in the results section as points of comparison.

4.1 Assumptions

For the CDAR protocol to work efficiently a few basic assumptions have been made about the network and node levels of hierarchy.

1. Clustered topology. Within each cluster there is a CH, one or more CMs, and one or more GNs, as in Figure 1.1.
2. Nodes deployed in the network are homogenous and have a unique ID.
3. Node positions are static and known a priori. The sink knows the locations of all the nodes in the network. Members know the locations of their neighbours. The CH knows the location of all the members within the cluster.
4. A single sink is present in the network and its location is fixed.
5. CHs can broadcast their data to their two-hop neighbours and unicast the data to their immediate neighbours. CMs can broadcast and unicast their data only to their immediate neighbours.
6. CMs can communicate with their neighbouring GNs and the CH, whereas the GNs can communicate directly only with their neighbouring GNs and CMs (using multihop routing to reach the CH).

7. Network size and the approximate number of clusters formed in the network should be known to the sink for timer synchronizations, so that the sink can transmit the parameters accordingly in the queries.
8. Network information is not collected and it is assumed that the parameters such as routing tables, cluster topology and node locations are established prior to any query.

4.2 General Solution

CDAR provides an end-to-end solution from contour detection and routing contour data to the sink after tracking the contour and performing in-network processing. Pattern-based contour detection is used to detect the contour. It provides the sink with the flexibility to repeat the detection process at a different location if the contour is not detected using the previous pattern. On detecting the contour, a cluster-based query propagation scheme is used to track the contour efficiently. The cluster-based scheme helps in making robust decisions from the member responses to track the contour and avoid erroneous readings. The cluster-based in-network processing schemes proposed help remove and reduce the redundant contour information. Finally, the contour information is routed to the sink in an efficient manner using the one of the proposed routing algorithms.

4.3 Querying Techniques

The important parameters transmitted in the query are the query ID, the contour value the sink is looking for, the pattern for contour detection, a minimum contour suppression threshold, look-ahead values for increasing the contour search area and also for finding broken contours and timer values for performing data routing to the destination. Some of these parameters are optional depending on the algorithm employed. For example, if the

suppression is disabled, then the suppression threshold is not transmitted in the query. The most common query propagation technique used in WSNs is flooding. However, flooding results in unnecessary wastage if the phenomenon is not uniformly spread throughout the network. Contour-based applications fall into this category. In this section, different query propagation techniques used for propagating the query in sensor based contour applications are explained in detail. Each of these techniques has its own advantages and disadvantages depending on the mission and network.

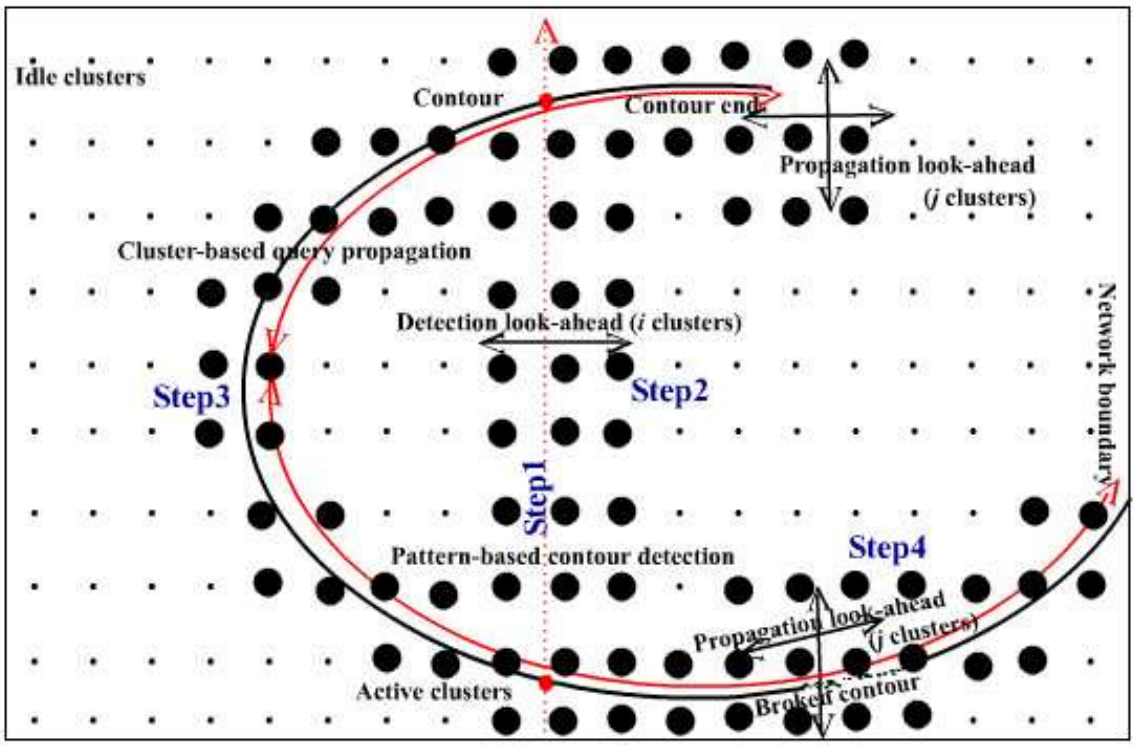

Figure 4.1: Overview of query techniques

There are two different phases in querying. The initial phase focuses on pattern-based contour detection techniques which detect the contour using a particular pattern. Single ray-based contour detection is shown in Figure 4.1. Detection techniques are associated with a detection look-ahead parameter which in this case indicates the number of clusters to look for perpendicular to the ray to query. The final phase deals with the actual query

propagation along the contour using cluster-based query propagation techniques. If the ray intersects the contour then the query is propagated along the contour in both the directions as shown in Figure 4.1. When queries propagated in opposite directions meet, the query propagation along the contour stops. To detect broken contours or natural contour ends, a propagation look-ahead parameter is used which indicate the number of clusters to look ahead before deciding if the contour is broken or terminated. In this section, known techniques are discussed for comparison along with new techniques which are part of the thesis contribution. Extensions to the new techniques for future work are also presented to establish the design extensibility.

4.3.1 Contour Detection Techniques

Contour detection is the initial phase of the querying technique. The main goal of these techniques is to route the query to detect a point on the contour in an efficient manner. As explained in the previous chapter, there are different contour detection techniques such as random walk, gradient routing and contour trees that can be used. All these approaches use information gain and communication cost to detect the contour efficiently. However, using these metrics may result in the query getting stuck at a saddle point, local minima or local maxima before the contour is detected. Moreover, the path traversed by these queries is not known at the sink, so the sink cannot efficiently re-transmit the query to a different location in the network, if a contour is not detected. In the detection techniques proposed, the query propagates along a geometrically specified pattern. The query is propagated as specified by the sink allowing retransmission of the query using a different pattern, if the contour is not detected.

4.3.1.1 Flooding

Flooding is the simplest and best known contour detection technique which propagates the query through the entire network. It guarantees detection of contours because every

node is visited. Flooding is presented here for a comparison with the proposed models. In flooding, a node that receives a query broadcasts it to all its neighbours. Nodes that receive the broadcast query further re-broadcast the query to their neighbours, if the query was not already received. In this manner, query is propagated through the network. In this thesis, flooding is performed to propagate the query through the network in a controlled manner. The sink transmits the request to the nearest CH which in turn broadcasts the request to its members indicating a set of GNs to forward it to the neighbouring CHs. None of the members broadcast the query request within the cluster other than the CH. Neighbouring CHs, on receiving the query, forward it to their members. CMs and GNs discard any queries received from neighbouring clusters if they have already received a query from their CH. In this manner, the query is propagated through the entire network. A detailed description of the cluster-based flooding algorithm is provided in Figure 4.2.

```
1.  CH:
2.  On receiving a CH query request:
3.      if request has already been received then
4.          discard the request
5.      else
6.          set request received to true
7.          broadcast the request to the members with the CH reading and GN ID list
8.          start a timer and wait for the query responses from the CMs/GNs
9.      end if

10. CM:
11. On receiving a CH query request:
12.     if request has already been received from CH then
13.         discard the request
14.     else if request is received from GN then
15.         forward the request to the CH
16.     else
17.         set query request received to true
18.         process the request
19.     end if

20. GN:
21. On receiving a CH query request:
```

```
22.  if request has already been received from CH then
23.      discard the request
24.  else if request is received from neighbouring GN then
25.      forward the request to the CH
26.  else
27.      set query request received to true
28.      if current node ID is in the GN ID list then
29.          forward the request to all the neighbouring CHs
30.      end if
31.      process the request
32.  end if
```

Figure 4.2: Flooding algorithm

Flooding is helpful if the phenomenon that the sink is interested in is spread uniformly throughout the network. For example, if the user is interested in the global maximum value, all the nodes must be queried. However, if the phenomenon is present in only a subset of nodes in the network then flooding is not efficient. For example, in contour applications the contour passes through a subset of nodes. Therefore, flooding propagates the request to the nodes with no relevant data, wasting system resources. Flooding is appropriate when there is unlimited power, a global solution requirement and if local information cannot constrain query space.

4.3.1.2 Single Pattern-Based Contour Detection

As seen in the previous section, flooding can be wasteful in contour-based WSN applications because the sensor nodes have limited power and the solution is a subset of the space. Some intelligent contour detection schemes are proposed which take the properties of the phenomenon into consideration and the WSNs constraints to provide an efficient solution. In the technique proposed, a pattern and detection look-ahead parameter is used in the query to detect the contour. The pattern in the query consists of the geographical location and the pattern shape. The geographical location in the query indicates from where the contour detection should start and the shape of the pattern tells how the routing needs be performed to detect the contour. A ray-based query pattern is

used to route the query to detect the contour in all the experiments for simplicity. However, other more complicated patterns can be used. The detection look-ahead parameter indicates the number of clusters on the either side of the pattern the query should be forwarded to broaden the contour detection area. There are three different kinds of query requests that are used by the algorithms to detect the contour. Initially, the *sink query request* is used by the sink to transmit a particular pattern. Clusters that lie on the pattern receive the sink query request and the CHs in these clusters broadcast the sink query request to their members, instructing the GNs forward the request to neighbouring clusters along the pattern. Next, the *forward CH query request* is broadcasted by the CHs that lie on the pattern to their members to propagate the query to their neighbouring clusters according to the detection-look ahead. CMs discard the received forward CH query request, but the selected GNs change the packet type to the *CH query request* and forward the request to the neighbouring clusters. The neighbouring clusters that lie on either sides of the pattern, on receiving the CH query request packet broadcast the request to their members. On member query response timer expiry, the CHs decrement the detection look-ahead and if it is valid, broadcast the forward CH query request and the procedure is repeated. For additional information regarding the query request packet structures, refer to the appendix.

This algorithm operates in two phases. In the initial phase, the query is routed along the clusters in a ray-based pattern using the direction contained in the query. Finally, the clusters that have received the query forward it to the neighbouring clusters in their vicinity based on the detection look-ahead parameter irrespective of whether they have detected a contour or not. The proposed ray-based contour detection algorithm assumes that the detection start location is from the sink as shown in Figure 4.3. However, the start of the detection can be set to any location in the network. A detailed description of the ray-based contour detection algorithm is provided in Figure 4.4.

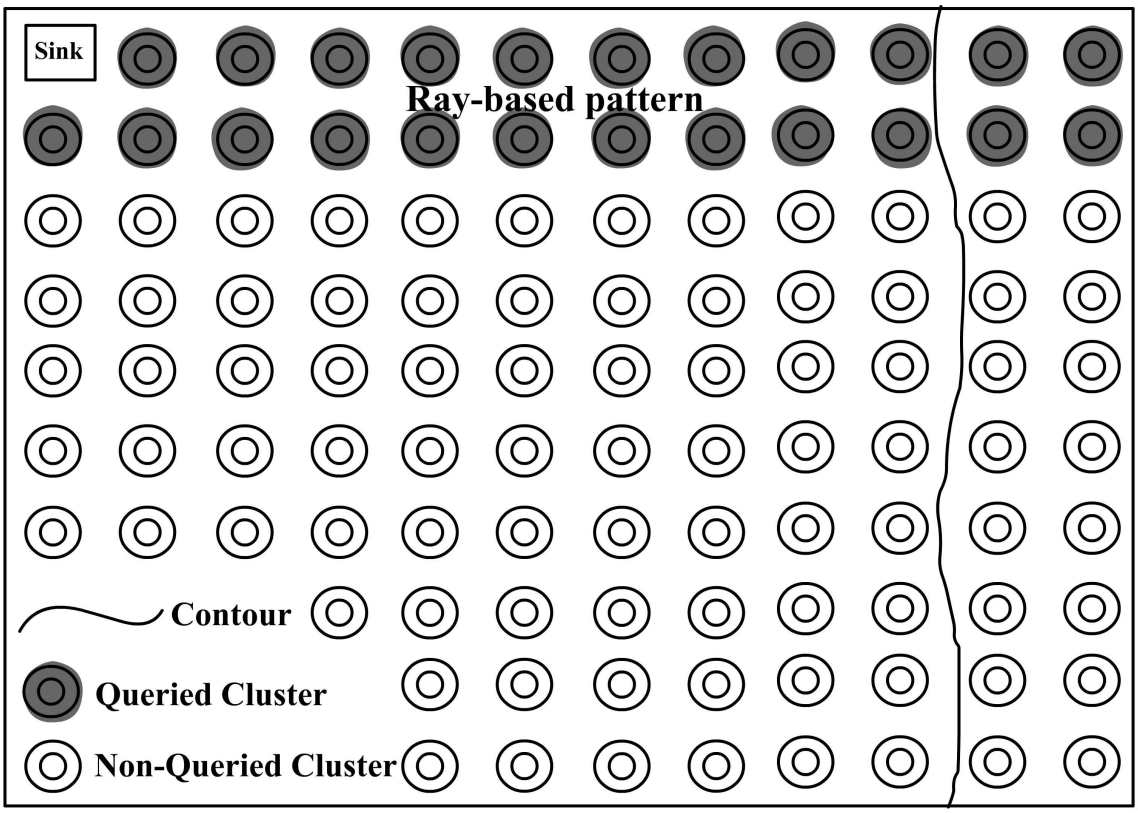

Figure 4.3: Single ray-based contour detection

```
1.  CH:
2.  On receiving a sink/CH query request:
3.      if request has already been received then
4.          discard the request
5.      else
6.          set request received to true
7.          if sink query request then
8.              set the detection look-ahead parameter to the user specified value
9.              broadcast the sink query request with the CH's reading and the GN ID
                list that needs to propagate the request to neighbouring CHs on the ray
10.         else
11.             broadcast the CH query request to the members with the CH's reading
12.         end if
13.         start a timer and wait for the query responses from the CMs/GNs
14.     end if
```

```
15. On query response timer expiry:
16.     if detection look-ahead is valid then
17.         broadcast the forward CH query request with the GN ID list
18.         decrement the detection look-ahead parameter value
19.     end if

20. CM:
21. On receiving a sink/CH query request:
22.     if request has already been received from the CH then
23.         discard the request
24.     else if request is received from GN then
25.          forward the request to the CH
26.     else
27.          set request received to true
28.          process the request
29.     end if

30. On receiving a forward CH query request:
31.     discard the request

32. GN:
33. On receiving a sink/CH query request:
34.     if request has already been received from the CH then
35.         discard the query request
36.     else if query request is received from neighbouring GN then
37.         forward the request to the CH
38.     else
39.         set query request received to true
40.         if sink query request then
41.             if current node ID is in the GN ID list then
42.                 forward the sink query request to the CHs on the ray if the GN
                    hasn't received any readings or responses from those clusters
43.             end if
44.         end if
45.         process the query request
46.     end if

47. On receiving a forward CH query request:
48.     if current node ID is in the GN ID list then
49.         change the forward query request to CH query request and forward the
            request to all those CHs that are in GNs vicinity from which the GN
            hasn't received any readings or responses
50.     end if
```

Figure 4.4: Ray-based contour detection algorithm

If some a priori estimate of the contour location is known, this approach can be effective in detecting the contour. In general, the contour location changes with time and the amount of change depends on the type of phenomenon being monitored. Taking these factors into consideration for selecting a proper pattern and look-ahead value helps in detecting the contour even if the contour location changes. If a contour is not found and the sink times out without a response from the network then a different pattern can be re-transmitted efficiently, as the path traversed by the previous pattern is known.

4.3.1.3 Multiple Pattern-Based Contour Detection

To address the obvious criticisms of single pattern-based contour detection two alternative scenarios are provided as extensions to the algorithm for cases when the approximate contour location is not known a priori. These algorithms are presented only, and are not implemented or characterized. The first of these algorithms is similar to the single pattern-based contour detection with the only difference being that the sink uses multiple patterns with different detection look-ahead values for these patterns in an attempt to intersect multiple contours or a single small contour with limited span across the network. These multiple patterns could be included in the same query or multiple queries can be used for each pattern. Multiple ray-based patterns using multiple queries for contour detection is shown in Figure 4.5. Ray2 and ray3 propagated by the sink have detected the presence of different disjoint contours, whereas ray1 doesn't detect any. Using this approach, there is still a chance of missing the contour if the contour is small and lies in between the rays. Where detecting contours of arbitrarily small size is involved, more exhaustive techniques must be employed by subdividing the space into sections smaller than the minimum allowed in the contour.

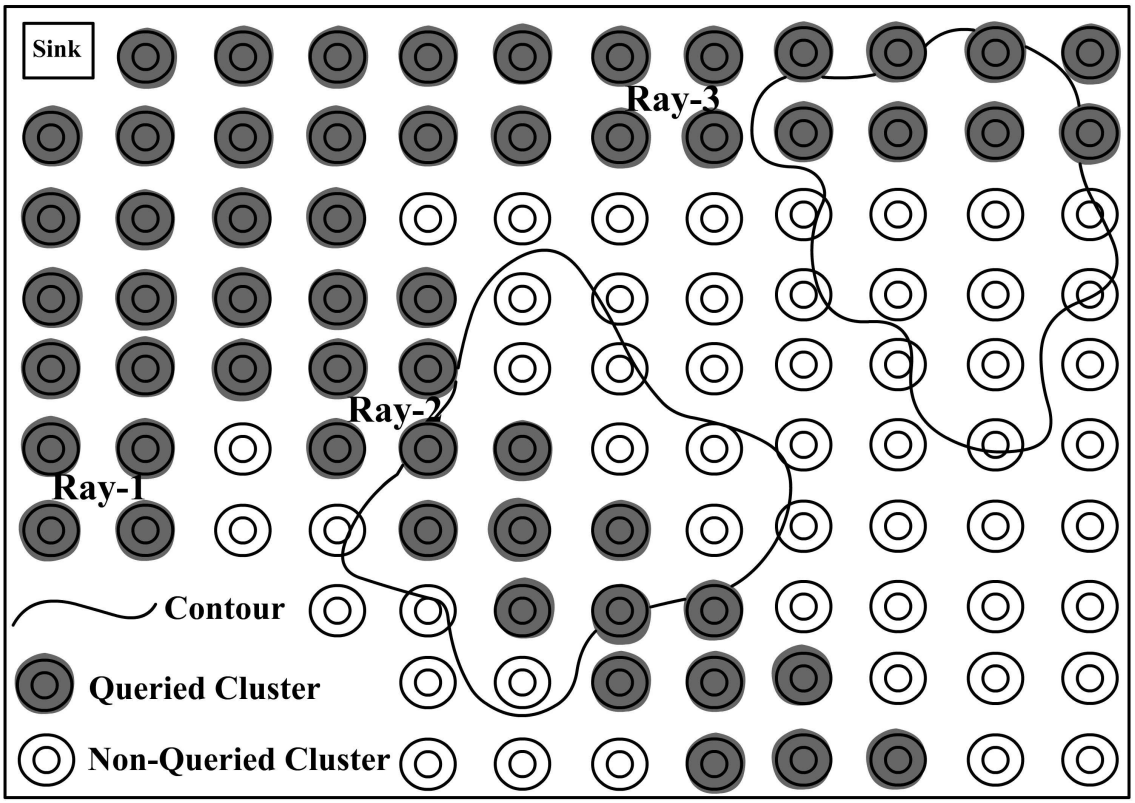

Figure 4.5: Multiple pattern-based contour detection

4.3.1.4 Raster Scan-Based Query Propagation

The raster scan-based propagation technique is used to exhaustively detect if a particular contour is present in the network. This technique is slightly different from the above two techniques in detecting the contour. It doesn't require any a priori knowledge of the contours in the network to perform detection. It uses a ray-based horizontal or vertical trace pattern to route the query among the clusters, as shown in Figure 4.6 for contour detection. Furthermore, detection look-ahead parameter is not used and the query is routed only among the clusters that lie on the pattern specified. This is to reduce the unnecessary communication overhead as the query is going to be propagated through the clusters until a contour is found. If the required contour is not present in the network, then it might be equivalent to or even better than flooding. But, if the contour is present in the

network then there can be considerable saving in propagating the query in this manner. Moreover, clustering ensures that the query doesn't get stuck by providing reliability from erroneous node readings and fault tolerance from single node failures.

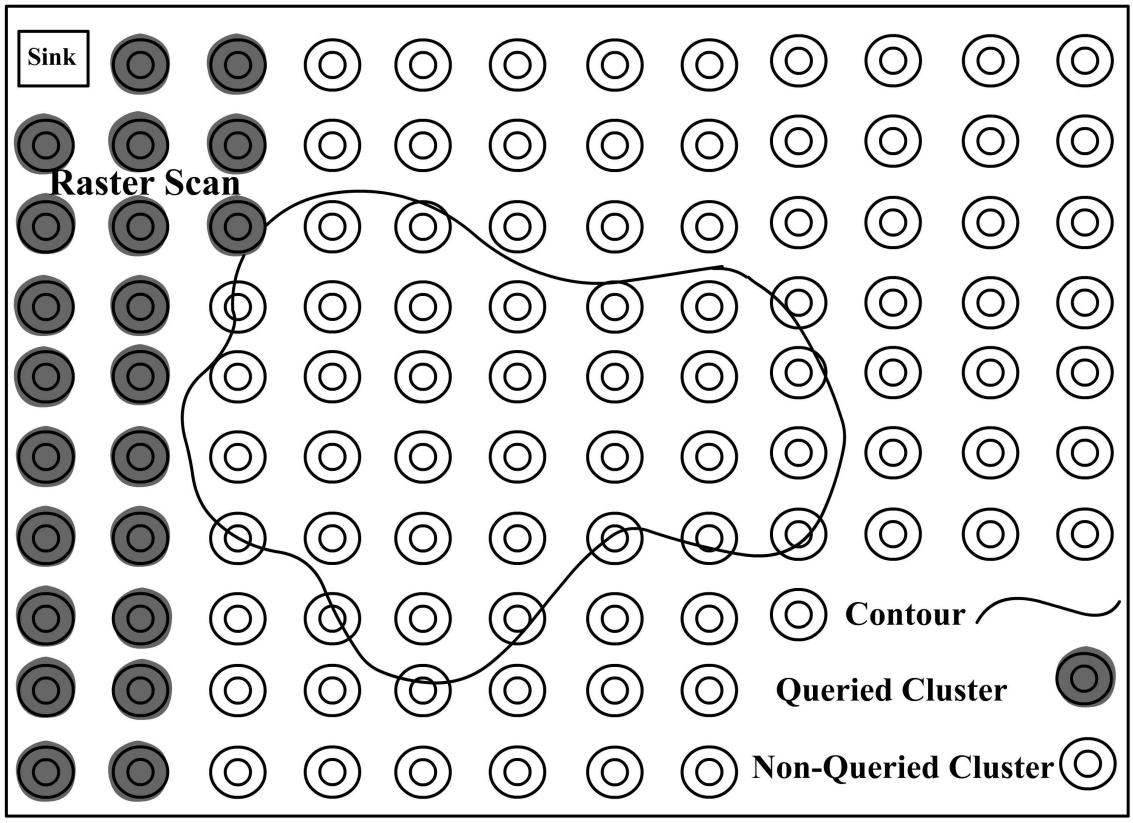

Figure 4.6: Raster scan-based contour detection

4.3.2 Query Propagation Technique

The query propagation phase is initiated once a point on the contour is detected using one of the proposed contour detection schemes described in the previous section. In this phase, the goal is to propagate the query in a reliable and efficient manner along the contour. Information gain and a propagation look-ahead parameter are used to propagate the query. Information gain at a particular node is based on the readings received from

the neighbouring nodes. A greater information gain helps make reliable decisions in choosing the nodes that should further propagate the query. The propagation look-ahead parameter is different from the detection look-ahead parameter, but provides similar functionality. In this section, cluster-based query propagation is explained in detail.

4.3.2.1 Cluster-based Query Propagation

The cluster-based query propagation technique is proposed to propagate the query along the contour on detecting a point on the contour as shown in Figure 4.7. The query propagation is done in both directions along the contour from the point of contour detection. A cluster-based scheme offers various benefits over the non-clustered schemes in propagating the query. In a non-clustered paradigm, there is no centralized entity to process the information and make decisions. The decision to select a node to propagate the query further is based on the information received from neighbouring nodes. These decisions may be incorrect if erroneous information is received from the neighbours. To avoid this problem, a cluster-based query propagation scheme is proposed where the decision making is performed at the CH based on the received member information. If a CH finds that a contour passes through the cluster, then a decision should be made in selecting the neighbouring cluster that has to further propagate the query.

To avoid this problem, the CH that has detected a contour propagates the query to all its immediate neighbouring clusters other than the originator of the query. Query propagation to the immediate neighbouring clusters increases the chance of tracking the contour accurately. The forward CH query request is used by the CH that has detected the presence of a contour in its cluster to propagate the query to the neighbouring clusters along the contour. Neighbouring cluster GNs on receiving the query check if they have already received a query from their CH. If so, they discard the received query otherwise they forward the query to their CH, which later broadcasts the CH query request to the cluster. In the querying phase, contour data is not propagated along the contour with the query because the path along the contour may not be the shortest path to the sink.

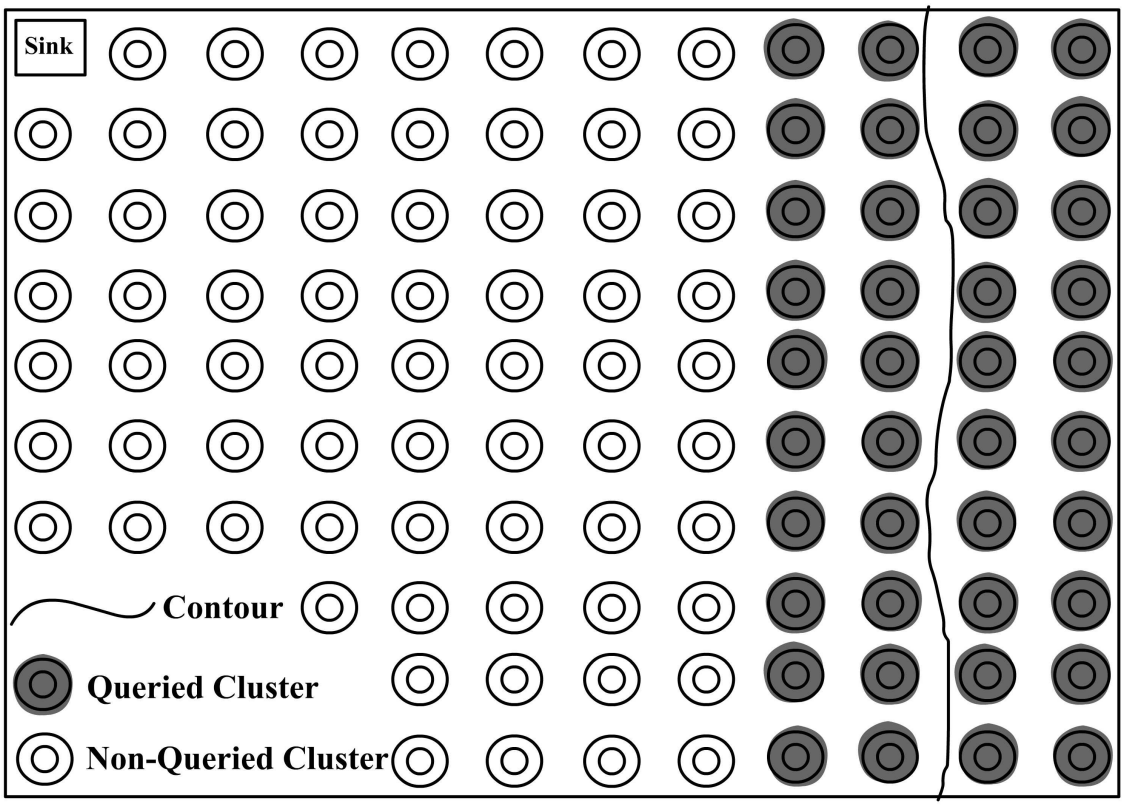

Figure 4.7: Query propagation along the contour after contour detection

Occasionally a contour can't be detected by the immediate neighbouring clusters. Moreover, detecting a natural contour termination is also important. To address all these problems and make the query propagation robust, a propagation look-ahead parameter is used, which broadens the search area by propagating the query to other clusters that are not the immediate neighbours of the current cluster. Based on the direction from which the query request was received, the current CH that has detected a contour indicates one of its immediate neighbouring clusters use the look-ahead and broaden the search, only if it doesn't detect any contour. If a closer look is taken into how the look-ahead parameter is used, it clearly shows that the parameter is not constant and varies depending on whether a contour is found or not. If a contour is found by the immediate neighbouring cluster, then the propagation look-ahead is zero. Otherwise, the search is broadened based

on the value set by this parameter. A detailed description of the cluster-based query propagation algorithm is provided in Figure 4.8.

```
1.  CH:
2.  On receiving a sink/CH query request:
3.      if request has already been received then
4.          discard the request
5.      else
6.          set request received to true
7.          if sink query request then
8.              broadcast the sink query request with the CH's reading and the GN ID
                list that needs to propagate the request to neighbouring CHs on the ray
9.          else
10.             broadcast the CH query request to the members with the CH's reading
11.         end if
12.         start a timer and wait for the query responses from the CMs/GNs
13.     end if

14. On query response timer expiry:
15.     if query responses are present then
16.         set the propagation look-ahead parameter to the user specified value
17.         indicate the CH ID that is in the contour direction to further propagate the
            query to its adjacent CHs, if a contour isn't detected
18.         broadcast the forward CH query request with the GN ID list
19.     else
20.         if CH ID in the CH query request is same as the CH ID then
21.             if propagation look-ahead value is valid then
22.                 decrement the propagation look-ahead value
23.             end if
24.             broadcast the forward query request with the GN list ID
25.         end if
26.     end if

27. CM:
28. On receiving a sink/CH query request:
29.     if request has already been received from the CH then
30.         discard the request
31.     else if request is received from GN then
32.         forward the request to the CH
33.     else
34.         set request received to true
35.         process the request
```

```
36.     end if

37. On receiving a forward CH query request:
38.     discard the request

39. GN:
40. On receiving a sink/CH query request:
41.     if request has already been received from the CH then
42.         discard the query request
43.     else if query request is received from neighbouring GN then
44.         forward the request to the CH
45.     else
46.         set query request received to true
47.         if sink query request then
48.             if current node ID is in the GN ID list then
49.                 forward the sink query request to the CHs on the ray if the GN
                    hasn't received any readings or responses from those clusters
50.             end if
51.         end if
52.         process the query request
53.     end if

54. On receiving a forward CH query request:
55.     if current node ID is in the GN ID list then
56.         change the forward CH query request to CH query request and forward
            the request to all those CHs that are in GNs vicinity from which the GN
            hasn't received any readings or responses
57.     end if
```

Figure 4.8: Cluster-based query propagation algorithm

4.4 In-network Processing Techniques

In sensor networks, the data is usually spatially and temporarily correlated. For most sensor applications, transmitting the raw data to the sink without in-network processing leads to wasted energy and can adversely impact the longevity of the network. To avoid redundant data being transmitted to the sink, spatially and temporarily correlated data is suppressed by in-network processing, allowing only the necessary data required by the sink to be transmitted. In-network processing can be done using aggregation and

compression techniques. Aggregation deals with removing redundant data using aggregate functions, suppression and response packet fusion. On other hand, compression reduces the size of the data that is to be transmitted by applying data encoding and decoding techniques. Even after performing aggregation and removing the redundant data, compression can be performed on the data to reduce the size of the data that is being transmitted. A combination of both these techniques provides an efficient in-network processing solution. In this section, various novel compression and suppression techniques proposed are discussed in detail and extensions to these techniques for future work are presented to establish design extensibility.

4.4.1 Compression Techniques

Compression techniques are used to efficiently encode the contour readings stored at a particular node as contour data in the query response payload. For example, transmission of contour location information to the sink can result in large data payload overhead and increases the energy cost in transmitting this information. The main focus of these compression techniques is to reduce the payload size by encoding only the appropriate information required by the sink to reconstruct the contour. As explained in the previous sections, suppression techniques only deal with suppressing the contour readings at a particular node and hence help save energy involved in transmitting these redundant readings. However, significant savings can also be obtained by applying compression techniques to reduce the payload size. The compression techniques proposed can be used independent of whether the suppression is enabled or disabled. In this section, detailed explanation of spatial compression and an overview of the temporal compression are presented.

4.4.1.1 Spatial Compression Algorithm

Spatial compression algorithms help in encoding the contour readings stored at a particular node efficiently as contour data in the query response payload. Nodes within a cluster on receiving the query request sense the external phenomenon and store the sensed reading. Later, nodes exchange their sensor readings with their neighbours in order to detect the presence of any contour in their vicinity. If a contour is detected from any of the received neighbour sensor readings, then the contour readings are populated and stored in the node. A contour reading consists of the node ID, neighbour node ID, location information and sensor readings. Encoding all these contour parameters increases the data overhead. The proposed spatial compression algorithm efficiently encodes the contour readings.

Given the assumption that the sink knows the node locations and a node in a cluster knows the location information of other members in the cluster, only encoding of the node IDs would be sufficient. The neighbour node ID is received in the sensed reading broadcast by the neighbour. It can also be used by the node to retrieve the stored neighbour node location information. Encoding the node IDs of the current node and its neighbour can also increase the data overhead depending on the size of the IDs. To reduce this overhead, every node's neighbour is assigned a relative node ID by the node and this information is also assumed to be known to the sink and other members within the cluster. Encoding the sensed readings by the nodes can also be costly. To reduce this overhead, the sensed readings are parameterized and the node's proximity to the contour is calculated using interpolation. The location parameter gives an approximate distance of the contour from the node based on an assumption of linear spatial variation between two nodes. For example, two neighbouring nodes of temperature 10 and 9 degrees can detect the presence of a 10 degree contour. In this case, the node which has a temperature of 10 degrees has a parameterized value of 0, whereas the neighbour node's value is 1. A value of 1 means the node is far away from the contour and vice versa. The final encoded data of each contour reading consists of the node ID, relative neighbour node ID and the node's proximity to the contour. The sink on receiving the encoded contour data in the

response decodes the data and reconstructs the contour map. A detailed description of the spatial encoding and decoding algorithms are provided in Figure 4.9 and Figure 4.10.

1. **CH/CM/GN:**
2. **if** contour is detected **at the node then**
3. **for** each contour reading stored at the node **do**
4. proximity = $\frac{\text{required contour value – node sensed reading}}{\text{neighbour sensed reading – node sensed reading}} \times 100\%$
5. encode the node ID, relative neighbour node ID and proximity distance
6. **end for**
7. **end if**

Figure 4.9: Spatial encoding algorithm

1. **Sink:**
2. **for** each decoded contour reading in the response **do**
3. retrieve the neighbour node ID from the relative neighbour node ID
4. proximity = proximity × 0.01
5. **if** node's proximity is 0 **then**
6. x = node's x-coordinate
7. y = node's y-coordinate
8. **else if** node's proximity is 1 **then**
9. x = neighbour's x-coordinate
10. y = neighbour's y-coordinate
11. **else**
12. d1 = calculate the distance between the node and its neighbour
13. d2 = calculate the adjacent edge distance from the node
14. angle = arccos(d1/d2)
15. neighbour's distance (to the contour point) = d1 × (1 – proximity)
16. x = neighbour's x-coordinate + (neighbour's distance × cos(angle))
17. y = neighbour's y-coordinate + (neighbour's distance × sin(angle))
18. **end if**
19. plot the x and y-coordinates of the contour location
20. **end for**

Figure 4.10: Spatial decoding algorithm

4.4.1.2 Temporal Compression Algorithm

Certain contour-based applications may require periodical observation of the phenomenon. Periodical observation of the phenomenon may result in the estimated contour position between two nodes approximately the same as in the previous observations. These observations are said to be temporally correlated. The degree of correlations between these observations depends on the type of phenomenon being monitored and the granularity interval between each successive observation. Temporal compression algorithms help in encoding these temporally correlated observations stored at a particular node efficiently as query response contour data payload. Details on detecting temporal correlations between the consecutive observations are explained in the temporal suppression algorithm section. Encoding of temporal data can be done using a contour-neighbour array specified by Cheng and Michael [44].

4.4.2 Suppression Techniques

Phenomena like temperature, pressure and humidity fields have high degree of spatial and temporal correlations. Contour-based WSN applications may generate highly correlated data. Transmission of this correlated data results in wastage of network resources. Efficient suppression techniques should remove these correlations in the sensed data. Moreover, these techniques should also provide the flexibility to the application to control the amount of suppression that is to be done. If this flexibility is not provided, then the user might not be interested in the data provided by the network, as it might lead to erroneous results. Suppression can be performed locally or globally in a distributed manner. Novel spatial and temporal cluster-based suppression algorithms are proposed in this section. Spatial suppression algorithms are discussed in detail, whereas temporal algorithms are presented for extensibility of the proposed suppression model and are part of future work.

4.4.2.1 Spatial Suppression Techniques

To provide a proper coverage of the signal field WSNs require dense node deployment. Due to high node density spatially proximate nodes have correlated readings. In a dense deployment, using spatial suppression techniques and suppressing unnecessary spatial correlated data helps in increasing the magnitude of energy savings by cutting down the cost involved in transmission of these extra bytes. This applies to contour-based WSN applications with dense node deployments where nodes that detect a contour have their readings spatially correlated. Moreover, spatial suppression techniques work efficiently with exploratory or one-shot queries. In this section, cluster-based spatial suppression algorithms are discussed in detail.

4.4.2.2 Suppression Logic

The proposed suppression logic is used internally by the cluster-based spatial suppression algorithm to remove the redundant sensed data. Suppression logic performs manipulation on the contour readings to remove redundant correlated data. A contour reading is constructed at a node when the node receives a sensed value reading from the neighbour and detects a contour between them. It can also be generated by decoding the contour data in the received query response from the neighbouring nodes. Contour data in the response packet is an encoding of multiple valid contour readings. Encoding of the contour data in the responses is described in detail in the compression techniques section. Each contour reading contains information about the IDs and locations of the node and its neighbour that have sensed the contour. Location information of the nodes is not transmitted in any of the messages, as it is costly. Instead, the location information of the nodes is retrieved based on node IDs, as the nodes IDs of the neighbours are known a priori. Nodes location information plays a crucial role in performing suppression. For performing cluster-based spatial suppression, two phases of the suppression logic need to be performed. In the first phase, the contour readings stored at the node are suppressed using the received query responses from the neighbouring nodes within the cluster. The received and buffered neighbour responses are decoded and each reading is compared

with the stored current node's readings. If the contour distance between the node's contour location and the decoded reading's contour location are within the minimum contour suppression threshold range, then the node's reading is discarded as shown in Figure 4.11. A suppression threshold is defined as the distance between the points on the contour and its value is propagated in the query to remove redundant correlated sensed data. Any records at a finer resolution than the suppression threshold are deemed redundant. After performing the suppression, the received neighbour responses are discarded unless the node is required to transmit a response to the CH.

In the second phase, the valid contour readings that are stored at the node are compared with each other and suppressed accordingly. Prior to performing the suppression, the node's contour readings are sorted based on the proximity between the nodes as shown in Figure 4.11.a. Next, the contour location is approximated from the node locations for each of the contour readings stored at the node as shown in Figure 4.11.b. Later, the contour distance is calculated between the contour location of the valid reading which has closest inter-nodal distance and the contour location of other readings as shown in Figure 4.11.c. Finally, the node's contour readings are compared with each other and if any of these readings lie within the threshold range they are discarded as shown in Figure 4.11.d. Sorting of the readings is done only once, but the other steps are repeated for the valid readings present. The suppression logic can also be extended to perform global distributed suppression. In that case, only the second phase of the core suppression logic needs be performed. The contour data present in each of the received responses from the lower layers is decoded and then all the decoded readings are sorted and compared with each other. However, the drawback of the global distributed approach is that the location information of the nodes that have detected the contour need to be transmitted with the contour data to perform suppression. Encoding the location information of the nodes is costly. A detailed description of suppression logic is provided in Figure 4.12.

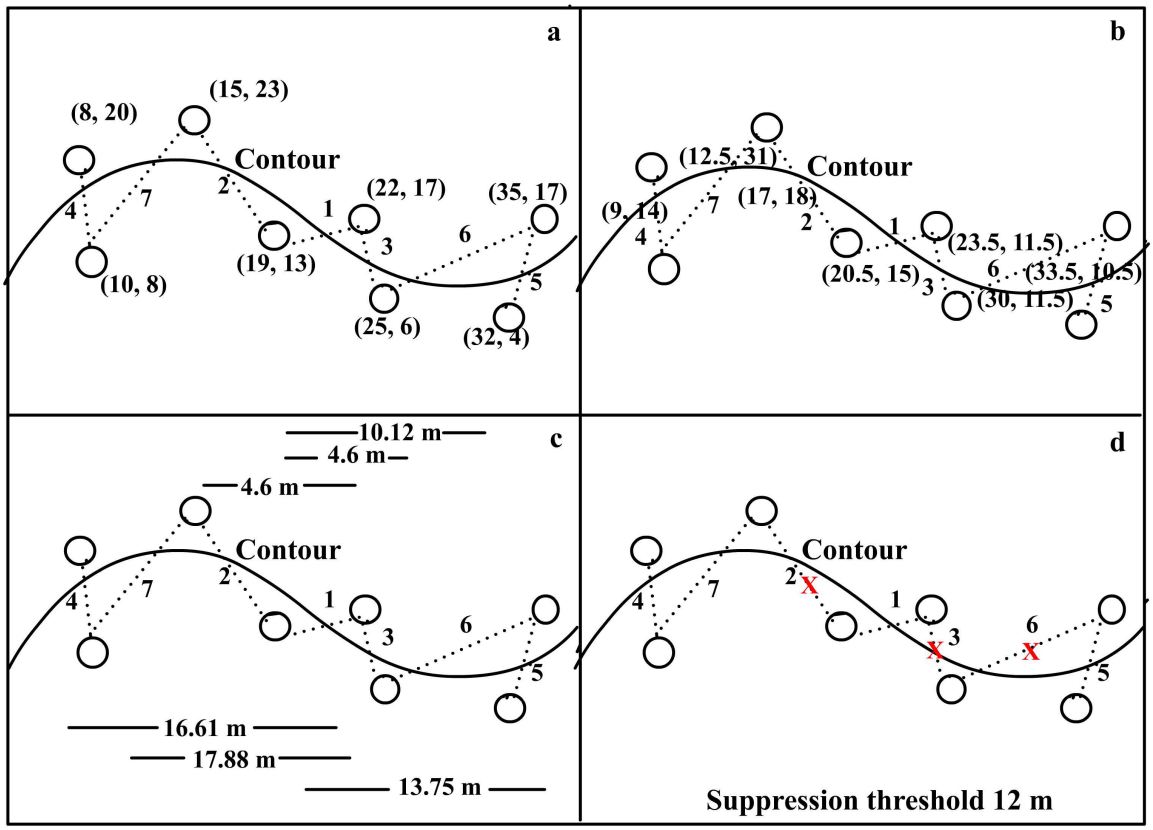

Figure 4.11: Illustration of the Suppression logic. a. Sort the contour readings received at a node based on the proximity between the nodes b. Approximate the contour location from the nodes for each reading c. Calculate the contour distance between the contour location of the reading that has the closest distance between the nodes (in this case it is indicated by 1) and the contour location of other readings d. Suppress all the readings (indicated by 'X') that are within the suppression threshold range

Phase 1:

1. **if** query responses are present in the packet buffer and contour readings are present in the data buffer **then**
2. **for** each query response in the packet buffer **do**
3. **for** each decoded reading1 in the query response **do**
4. **for** each reading2 in the data buffer **do**
5. **if** valid reading2 in the data buffer is within the minimum contour distance suppression threshold of reading1 **then**
6. set the reading2 in the data buffer to invalid
7. **end if**
8. **end for**

```
9.          end  for
10.         discard the query response from the packet buffer
11.     end for
12. end if

Phase 2:
1.  if readings are present in the data buffer and packet buffer is empty then
2.      sort the readings in the buffer based on the closest distance between the nodes
3.      for each reading1 in the data buffer do
4.          for each reading2(reading1 + 1) in the data buffer do
5.              if valid reading2 in the data buffer is within the minimum
                contour distance suppression threshold of reading1 then
6.                  set the reading2 in the data buffer to invalid
7.              end if
8.          end for
9.      end for
10. end if
```

Figure 4.12: Suppression logic

4.4.2.3 No-Suppression Algorithm

The no-suppression algorithm is a baseline algorithm implemented for comparison with the proposed spatial suppression algorithms. It doesn't perform the actual suppression of the contour data or control messages. An understanding of the design and logic of this algorithm helps in appreciating how the suppression algorithms work and their impact. The proposed suppression and no-suppression algorithms are based on a clustered network topology. According to Pattem *et al.* [43] a clustered topology with optimal cluster size would perform well for a wide range of spatial correlations. These proposed suppression and no-suppression algorithms can also be extended to multi-hop clusters of size greater or lesser than two hops (the standard size in the experiments) and to randomly distributed clustered topologies.

In the proposed no-suppression algorithm, the query request broadcast by the CH is assumed to be received by all the members within the cluster, so that the members need not re-broadcast the query request internally. To broadcast the query request to all the

members, the CH needs greater power and the amount of power needed varies with the cluster size. However, the suppression and no-suppression algorithms are not impacted even if the CH broadcasts the query request to its one-hop neighbours and they further re-broadcast it to the other members. In the no-suppression algorithm, member responses are aggregated only at the CH before routing the overall aggregate response to the destination. Taking the memberships of the nodes in a cluster into consideration, two variants of this algorithm is developed. One variant is for the CH and the other for the CMs/GNs.

In this approach, the CH on receiving the query request broadcasts the request within the cluster with its sensed reading along with the contour information it is looking for as shown in Figure 4.13.a. On receiving the request from the CH, CMs that are one-hop from the CH check if a contour exists between themselves and the CH. If so, the CMs populate the contour reading between themselves and the CH and store the reading. On the other hand, GNs and CMs that are more than one-hop away don't populate any contour readings from the received CH query request even if a contour exists because an accurate contour location might be difficult to obtain because of the distance. Nodes in the cluster start a query response timer on receiving the query. The response timer value varies depending on the membership of the node. The CMs/GNs have a similar response timer value in the no suppression algorithm and their timer value is less than the CH. Later, CMs and GNs within the cluster broadcast their sensed readings to their neighbours as shown in Figure 4.13.b. Neighbours on receiving the sensed readings check if a contour exists between the node that broadcast the sensed reading and itself. If so, a contour reading is generated and stored at the node. On the other hand, even though CHs detect a contour between themselves and the broadcast CM sensed reading, they don't populate any contour reading because CMs that are one-hop away would have populated the contour readings from the CH sensed reading broadcast in the query request. This avoids unnecessary duplication of the contour data at the CH.

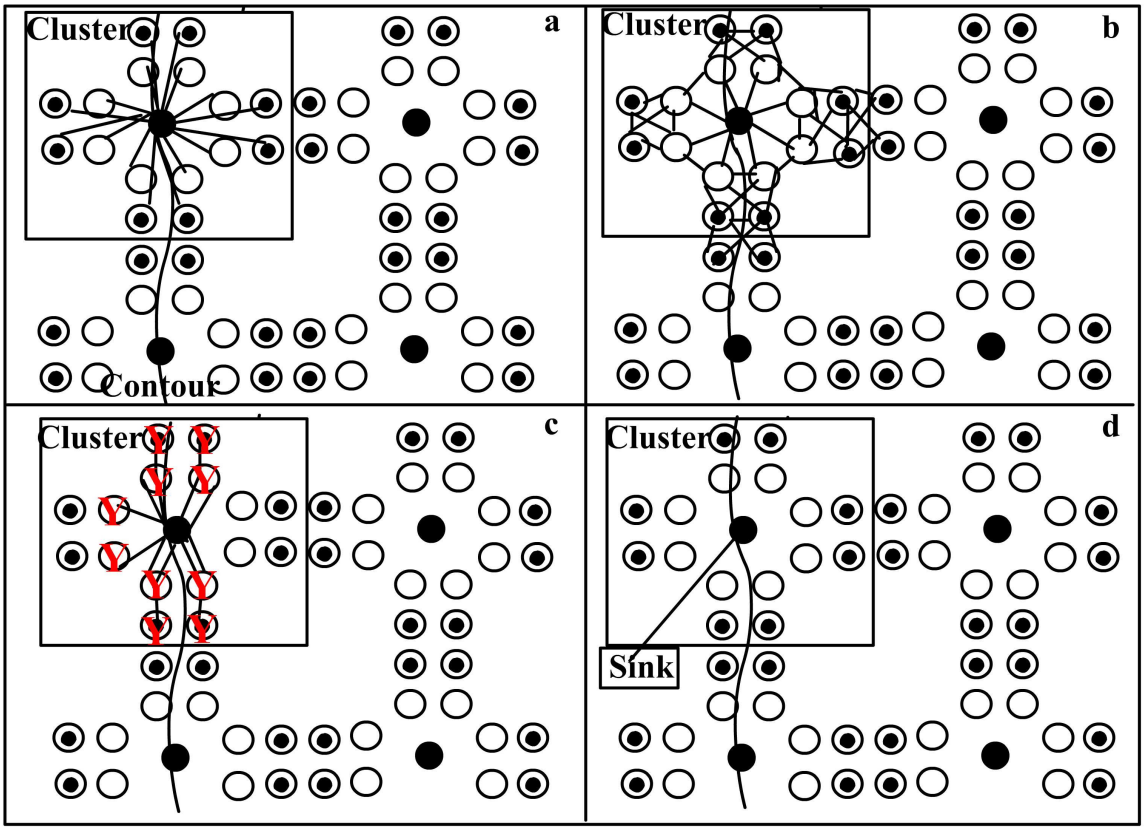

Figure 4.13: Illustration of the No-suppression algorithm. a. Query request broadcast by the CH to its members b. Sensed readings broadcast by the members to their neighbours c. Query responses are unicast by the members (indicated by 'Y') that have detected the contour to their CH d. Overall aggregated members response forwarded by the CH to the sink

There are two different scenarios that can occur based on whether a contour passes through the cluster or not. In the first scenario, assume that the contour doesn't pass through the cluster. All the members exchange their sensed readings with the neighbours on receiving the query request. On response timer expiry, none of the members transmit any messages or contour data to the CH. The CH response timer times out and moves to idle state confirming that the contour doesn't pass through the cluster. In the second scenario, assume that the contour passes through the cluster. Like the previous scenario, the members exchange the sensed readings with each other. Members that have detected the contour and have contour readings stored encode the readings in the response as contour data and forward the response to the CH independently using unicast as shown in

Figure 4.13.c. On response timer expiry, the CH aggregates all the responses into a single response and forwards the overall aggregate response to the sink based on the routing algorithm.

This approach provides the most complete data about the contour, as the sensed data is transmitted without any processing. Even with suppression disabled, the proposed algorithm avoids unnecessary transmission of query request and response control messages within the cluster. It also provides a mechanism to aggregate all the member responses received at the CH into an overall aggregate response and in the process strips the individual member response packet headers before routing the overall response to the destination. However, the problem with this approach is that the data might be redundant for dense networks and may result in unnecessary energy wastage. Moreover, this approach doesn't give the application the flexibility to control the amount of suppression. To counteract these problems and provide an efficient suppression of contour data and control messages, cluster-based spatial suppression algorithms are proposed in the next section. A detailed description of the no-suppression algorithm is provided in Figure 4.14.

```
1.  CH:
2.  On receiving CM/GN query response:
3.      if response is forwarded to the destination then
4.          discard the received response
5.      else
6.          store the received responses in the packet buffer
7.      end if

8.  On receiving CM/GN sensed reading:
9.      discard the received reading
10.
11. On  query response timer expiry:
12.     aggregate and encode the readings as contour data in the response

13. CM/GN:
```

```
14. On receiving sink/CH query request:
15.     if contour exists between CM (one-hop from CH) and CH then
16.         populate and store the contour reading in the data buffer
17.     end if
18.     broadcast the CM/GN sensed readings
19.     start a query response timer at the CM/GN and wait for the sensed readings

20. On receiving CM/GN sensed reading:
21.     if query response is forwarded to the CH then
22.         discard the received reading
23.     else if contour exists between CM/GN and received sensed reading then
24.         populate and store the contour reading in the data buffer
25.     else
26.         discard the received reading
27.     end if

28. On  query response timer expiry:
29.     aggregate and encode the readings as contour data in the response
30.     forward the query response to the CH independently
31.     set the query response sent to true
```

Figure 4.14: No-suppression algorithm

4.4.2.4 Cluster-Based Spatial Suppression Algorithm

The cluster-based spatial suppression algorithm is designed to suppress redundant contour data. The CH is the central entity within a cluster that takes all the critical decisions. In all the proposed suppression algorithms the CH aggregates all the received member responses before forwarding the data to the destination. In the no suppression algorithm, members within the cluster forward their responses by unicast to the CH independently, on response timer expiry. This can result in energy wastage, if the cluster size is large. To avoid this waste of energy, the proposed suppression algorithms forward the responses to the CH in a controlled manner. Moreover, a minimum contour suppression threshold parameter is used to control the amount of suppression that is to be performed by a particular member. The value of this parameter is indicated by the sink in the query request and a larger value indicates more widespread suppression. More details on how this parameter is used to perform suppression are explained in the suppression logic section.

In the proposed suppression algorithm, the cluster is treated as an aggregation tree with the CH as the root of the tree. All the CMs act as intermediate nodes in the aggregation tree and the GNs act as the leaf nodes of the tree, as they lie on the cluster boundary. Members in the cluster have different response timer values and this value varies depending on the hop count to the CH. Those members closer to the CH have a larger timer value, as the members move away from the CH the timer value decreases. The response timer value of the CH is larger than that of any member's response timer value. There is a slight difference in the way the response timer is set by the CMs. The response timer values of the CMs that are on the same hop level and that haven't detected any contour from the broadcast sensor reading in the query request by the CH have a greater timer value compared to the CMs that have detected a contour. The reason for the response timer value difference between the CMs within the same hop level is because CMs exchange their sensed reading with their neighbours only when they detect a contour. From these broadcast sensed readings, the neighbour CMs within the same hop level can detect the presence of a contour between them and the CM that has broadcast the response. On the other hand, all the GNs at a particular hop level have a similar response timer value, because all the GNs broadcast their sensed readings to their neighbouring nodes separately, unlike the CMs which broadcast their sensed reading in the response to the CH. GNs broadcast their sensed readings to their neighbours separately in order to enable the neighbouring cluster GNs to suppress their query responses. In this case, it is not possible to detect a contour between the GNs and its one-hop CMs, if the GNs didn't broadcast their sensed readings or query responses.

The advantage of using a cluster-based aggregation tree approach is contour readings suppression and contour data aggregation can be performed simultaneously at the members as the responses propagate towards the CH. Contour readings are populated and stored by the members from the broadcast neighbours sensed readings only when a contour is detected. These readings are encoded into the query response as contour data using the encoding scheme mentioned in the compression techniques section.

Suppression of redundant contour readings at a member means that the actual redundant contour data is removed. As explained in the suppression logic, a member suppresses the contour readings in two phases. These suppressed readings can be inter-cluster or intra-cluster contour readings. In the first phase, if the members have any received broadcast member query responses, they decode the contour data in these responses and each of the decoded reading is compared with its stored contour readings, if present. If any of the member's stored readings is within the minimum contour suppression threshold then the reading is suppressed. In the second phase, the stored contour readings are sorted based on each reading's proximity between the nodes and compared within themselves. If any of these readings are within the minimum threshold limit they are suppressed.

Apart from suppression, efficient contour data aggregation can also be performed using the proposed cluster-based aggregation tree approach. A member whose response timer expires and that has detected a contour, performs suppression before it broadcasts the response to its one-hop neighbours. One of the neighbours is designated to forward the data further. On receiving the broadcast query response, members store the received query response when their response timer expires. Members that are not designated to forward the contour data discard the received query response after performing suppression. However, the member that is designated to forward the query response aggregates the contour data received to its own stored data and propagates the packet further. This process continues until the response reaches the CH. Members that have already forwarded the response discard any received responses. By performing suppression and aggregation at the intermediate nodes instead of the CH, unnecessary redundant data transmissions within the cluster are avoided. On receiving the aggregated responses from the members, the CH decodes the contour data present in these responses and stores the contour readings. Only the second phase of the member suppression algorithm is applied to these stored readings.

The spatial suppression algorithm also suppresses control messages within the cluster. Query request message transmissions by the members and broadcasting of the CMs

sensed reading are the control messages that can be suppressed. Query request suppression by the members is the same as in the no suppression algorithm. In general, all the nodes exchange their sensed readings with their neighbours to detect the presence of a contour. However, with the spatial suppression algorithm, most of the CMs refrain from transmitting their sensed reading to their neighbours even if they detect the contour present in their vicinity. The CMs that are one-hop from the CHs or GNs can receive the broadcast reading from the CH and the GNs in a multi-hop cluster. From these broadcast readings, CMs can detect if a contour exists between themselves and the broadcaster. Those CMs that have detected a contour based on the received sensed readings from the CH or GNs include their sensor reading in the query response while forwarding the response to the CH using broadcasting. On receiving the CM query response, other CMs compare their reading with the sensed reading to detect the presence of a contour. If a contour is detected, CMs include their reading in the response and broadcast the packet to their neighbours.

On receiving a query request, the CH forwards the suppression threshold interval, contour value and its sensed readings to the members as shown in Figure 4.15.a. All the members on receiving the query request start their query response timers based on whether they have detected a contour between themselves and the CH. The GNs broadcast their sensed readings to their neighbours on receiving the request as shown in Figure 4.15.b, which store them if a contour exists between them and the GN that broadcast the reading. On response timer expiry, members broadcast their responses to their one-hop neighbours and designate one of the neighbours that are closer to the CH to forward its contour data further. On receiving the query response broadcast, members suppress their contour readings by applying the suppression logic and discard the received query response if they are not designated to propagate the contour data in the response further. The member that is designated to propagate the response further as shown in Figure 4.15.c aggregates the received contour data in the response with its contour data and broadcasts the overall aggregated response to its neighbours as before. In this manner, the response is propagated until the CH receives the data as shown in Figure 4.15.d. At the CH, the final level of suppression is performed on all the received data before the readings are

aggregated into an overall response and forwarded using one of the schemes described in section 4.5.

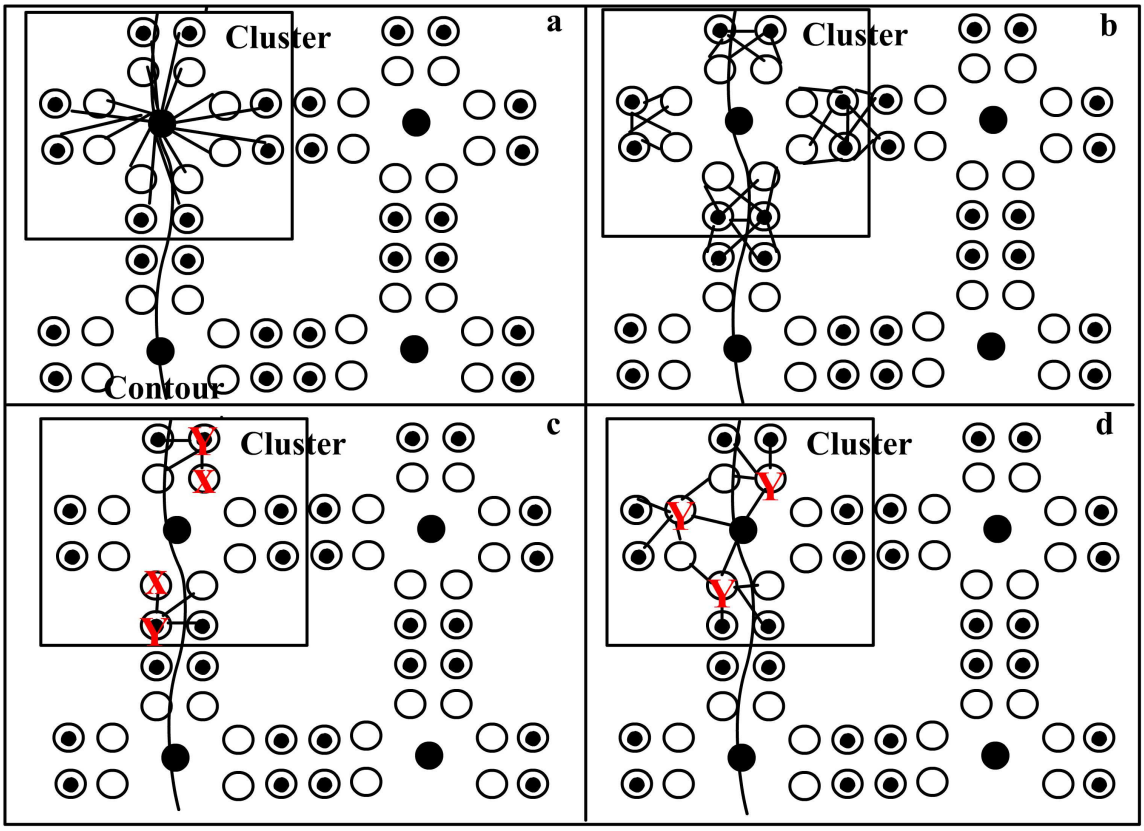

Figure 4.15: Illustration of the Cluster-based spatial suppression algorithm. a. Query request broadcast by the CH to its members b. Sensed readings broadcast by the GNs to their neighbours c. Query responses broadcast by the GNs (indicated by 'Y') that have detected the contour to their neighbours indicating the forwarding CM (indicated by 'X') after performing suppression d. Query responses broadcast by the CMs (indicated by 'Y') that have detected the contour to their neighbours after performing suppression.

In one scenario, it is assumed that the contour doesn't pass through the cluster. In this case, only the GNs exchange their sensed readings with the neighbours on receiving the query request, whereas none of the CMs broadcast any of their sensed readings. The savings is the suppression of CM readings broadcasts. On response timer expiry, none of the members transmit any messages or contour data to the CH. In the second scenario, it

is assumed that the contour passes through the cluster. Like the previous scenario, the GNs exchange their sensed readings with each other and only the CMs that have detected the contour broadcast their readings while other CMs are suppressed, saving unnecessary broadcasts by the CMs that haven't detected the contour. Members that have detected the contour perform suppression and aggregate all the received responses at each hop while propagating the response to the CH. This removes the redundant contour data and saves energy by cutting down unnecessary data transmissions. Moreover, the members aggregate the responses at each hop and send the overall aggregated response to the CH avoiding independent transmission of responses by the members. The amount of suppression that is to be performed at a particular member is controlled by the minimum contour suppression threshold parameter broadcast in the query.

Another important parameter that can be introduced for performing suppression along the contour when the suppression threshold spans more than a cluster is the use of a skip-ahead parameter. Clusters that have detected a contour forward this parameter in the request packet to the neighbouring clusters. On detecting a contour, the neighbouring clusters look into the skip-ahead parameter and suppress their responses if they are within the threshold limit. This technique is not implemented in the thesis and is part of future work. A detailed description of the cluster-based spatial suppression algorithm is provided in Figure 4.16.

```
1.  CH:
2.  On receiving CM query response:
3.      if response is forwarded to the destination then
4.          discard the received response
5.      else
6.          decode  the contour data and store the contour readings in the data buffer
7.      end if

8.  On receiving CM/GN sensed reading:
9.      discard the received reading

10. On  query response timer expiry:
```

```
11.  perform suppression on the stored contour readings using suppression logic
12.  if valid contour readings are present then
13.      aggregate and encode the readings as contour data in the response
14.  end if

15. CM:
16. On receiving sink/CH query request:
17.  if contour exists between CM (one-hop from CH) and CH then
18.      populate and store the contour reading in the data buffer
19.  end if
20.  start a query response timer at the CMs based on the hop distance to the CH
     and detection of a contour

21. On receiving GN sensed reading:
22.  if contour exists between CM and received sensed reading then
23.      populate and store the contour reading in the data buffer
24.  else
25.      discard the received reading
26.  end if

27. On receiving CM/GN query response:
28.  if query response is forwarded to the CH then
29.      discard the received response
30.  else
31.      if contour exists between CM and the sensed reading received in the
         CM/GN response then
32.          populate and store the contour reading in the data buffer
33.      end if
34.      if CM ID is same as the transmit ID in the response then
35.         decode  the contour data and store the contour readings in the data buffer
36.      else
37.         store the received response in the packet buffer
38.      end if
39.  end if

40. On  query response timer expiry:
41.  perform suppression on the stored contour readings using suppression logic
42.      if valid contour readings are present then
43.          aggregate and encode the readings as contour data in the response
44.          broadcast the response to the neighbours indicating the transmit ID
             that needs to further propagate the contour data to the CH and also
             the CM's sensed reading
45.      end if
46.  set the query response sent to true

47. GN:
```

```
48. On receiving sink/CH query request:
49.     broadcast the GN sensed readings
50.     start a query response timer at the GNs based on hop distance to the CH

51. On receiving GN sensed reading:
52.     if contour exists between GN and received sensed reading then
53.         populate and store the contour reading in the data buffer
54.     else
55.         discard the received reading
56.     end if

57. On receiving GN query response:
58.     if query response is forwarded to the CH then
59.         discard the received response
60.     else
61.         store the received response in the packet buffer
62.     end if

63. On  query response timer expiry:
64.     perform suppression on the stored contour readings using suppression logic
65.     if valid contour readings are present then
66.         aggregate and encode the readings as contour data in the response
67.         broadcast the response to the neighbours indicating the CM transmit ID
            that needs to further propagate the contour data to the CH
68.     end if
69.     set the query response sent to true
```

Figure 4.16: Cluster-based spatial suppression algorithm

4.4.2.5 Temporal Suppression Techniques

Some contour-based applications may require observing the phenomenon periodically and reporting the information to the destination. Periodical observation of the phenomenon may result in the estimated contour position between two nodes approximately the same as in the previous observations. These observations are said to be temporally correlated. In such cases, applying only spatial suppression techniques might not be efficient because even after spatial suppression the reported contour data at the destination might be redundant due to temporal correlations. To remove redundant temporally correlated contour data the proposed cluster-based temporal suppression

algorithm can be used. The temporal suppression algorithm presented here is for extensibility of the proposed suppression model and is part of future work.

Cluster-Based Temporal Suppression Algorithm

The main focus of cluster-based temporal suppression algorithm is to detect and suppress the redundant temporally correlated contour data in contour-based WSN applications which require periodical observation of the phenomenon. It can be used in conjunction with spatial suppression algorithms. Temporal suppression of correlated data is performed at two levels, one level at the members and the other at the CH. Members that observe and detect the phenomenon for the first time store their contour readings after performing spatial suppression. Similarly, the CH decodes the contour data received in the responses from the members and stores the readings after performing spatial suppression and forwarding the overall aggregated response to the destination. Temporal suppression is not performed for the first time by the CH and members when they observe and detect the phenomenon. From the next observation and detection of the phenomenon, the members compare their contour readings with those readings stored previously, and if any of the observed readings are temporally correlated, then they are suppressed. A temporal suppression threshold can be used to control the amount of temporal suppression by specifying the correlation degree for suppression.

Temporally correlated contour readings are not suppressed completely by the members because the CH has no information on whether the members have actually suppressed their contour readings by performing temporal suppression or there was no contour passing through the cluster. To avoid this problem, those members that have detected a contour and have their readings temporally correlated, encode minimal information in the response to the CH. If some contour readings that are detected by the members are different from those stored previously, then they are stored by the members along with their existing readings. The CH, on receiving the responses from its members, performs another level of temporal suppression to remove any further redundant readings. The

savings in temporal suppression are obtained by encoding minimal data in the response payload while forwarding the response to the CH or to the sink. Each of the stored readings at the members and the CH is associated with a time duration for which the readings are valid. This duration is indicated in the query request propagated by the sink. If the newly observed contour readings are temporally correlated with those stored previously in the node then their duration is re-initialized. The duration should be chosen based on the frequency the members observe the external phenomenon and the type of phenomenon being observed.

4.5 Contour Data Routing Algorithms

Data routing is the final phase of any sensor network application and it involves transmitting the information requested in the query to the sink. Efficient data transmission to the sink is important in sensor networks because energy is constrained. In this section, different data routing techniques used for transmitting the data to the sink in contour-based WSN applications are analyzed in detail. Routing of the contour data is dependent on the query propagation mechanism. All the proposed routing techniques fall in two classes: independent shortest path routing, and tree-based routing. In the shortest path routing, the CHs that have the contour information transmit the responses to the sink along the shortest path independently. In the tree-based routing, the CHs that have contour information transmit the response to the sink along the shortest path along the tree. The advantage of the tree-based routing approach is in-network processing can take place at the intermediate nodes as the responses propagate the along the tree to the destination. All the proposed algorithms use a cluster-based topology. Flooding-based Shortest Path Routing (F/SPR) and Information-driven Shortest Path Routing (I/SPR) use independent shortest path routing to route the data to the destination. Flooding-based Aggregation Tree-based Routing (F/ATR), Information-driven Aggregation Tree-based Routing (I/ATR) and Information-driven Contour and Aggregation Tree-based Routing (I/CATR) use the tree-based routing to route the responses to the destination. Information-driven Contour and Shortest Path Routing (I/CSPR) is an extension routing

algorithm presented for the completeness of the design and is part of the future work. F/SPR and F/ATR algorithms are the baseline algorithms implemented for comparison. I/SPR, I/ATR, I/CATR and I/CSPR are the novel algorithms proposed here. Though I/SPR and I/CSPR are present in the literature [31-34], use of clustering as the underlying topology makes them different from the existing contour-based data routing algorithms. Table 4.1 shows the list of algorithms that are designed and the features on which they are based.

Table 4.1: Algorithms overview

	Flooding	**Ray/Contour**
Shortest path	F/SPR	I/SPR
Aggregation Tree	F/ATR	I/ATR
Contour + Aggregation tree		I/CATR
Contour + Shortest path		I/CSPR

4.5.1 Flooding-based Shortest Path Routing

F/SPR is a baseline shortest path algorithm implemented for comparison with the proposed algorithms. F/SPR algorithm is the simplest of all the routing approaches described in this thesis. In this algorithm, all the nodes in the network receive the propagated query request by the sink. On receiving the query, all the nodes in the network sense the external phenomenon and exchange the sensed readings accordingly to detect the presence of a contour. Nodes in the cluster that detect the contour inform their CH. The member responses received by the CH are aggregated and the overall aggregated response is routed to the sink through the shortest path independently as shown in Figure 4.17.

The advantage of F/SPR algorithm is there isn't much delay in routing the response to the sink because this approach doesn't require any synchronization between the clusters. Clustering reduces some overhead by aggregating the individual node responses at the

CH and allowing only the CHs to transmit the response to the sink. Even with the CHs transmitting the data, there is overhead incurred because the individual response headers are forwarded at each hop. While F/SPR is the most general approach, it misses several opportunities for application-specific data aggregation. Moreover, the query propagation through the network using flooding is not efficient for contour-based applications. To avoid unnecessary transmission of individual response headers, tree-based approaches described in the following sections can be employed.

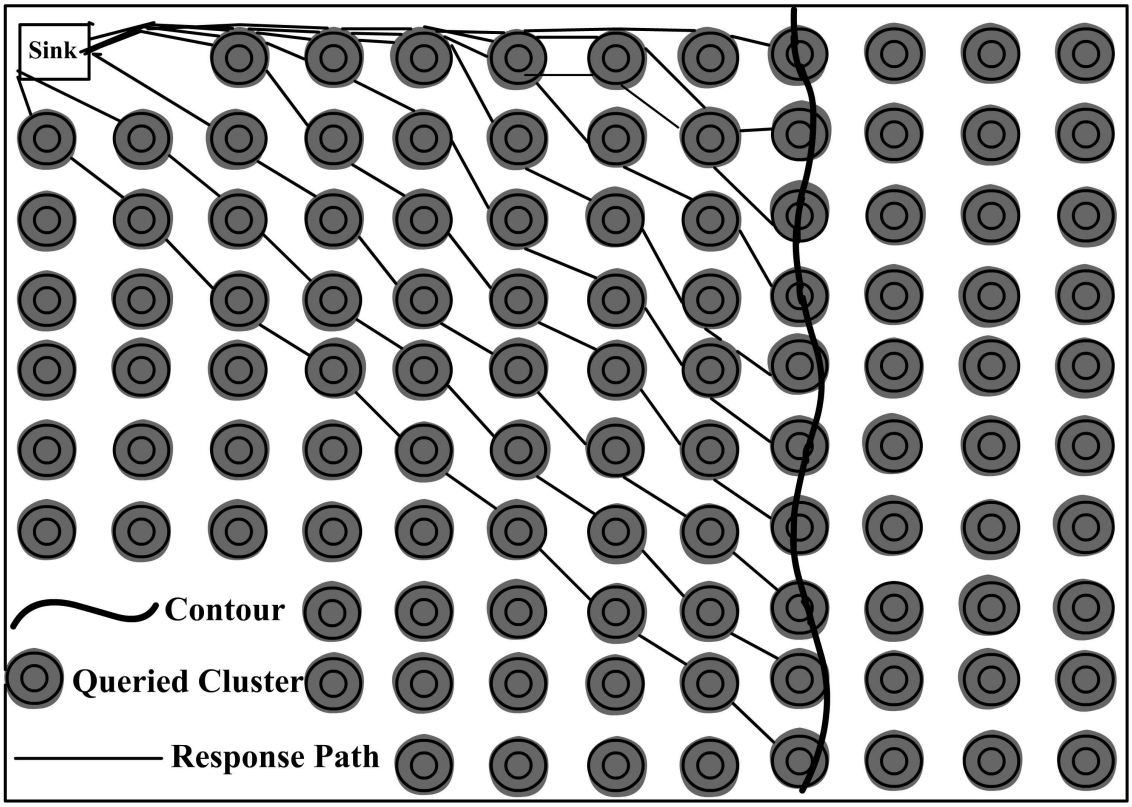

Figure 4.17: Illustration of the F/SPR algorithm

4.5.2 Flooding-based Aggregation Tree-based Routing

F/ATR is a baseline tree-based shortest path algorithm implemented for comparison with the proposed algorithms. Like in F/SPR, the query needs to be flooded in the network for

all the nodes to start the aggregation timers to schedule the propagation of responses from lower layers to higher layers. In F/ATR algorithm, the query responses from the lower layers get aggregated at the higher layers of the tree while routing the query responses to the sink in the shortest path along the tree as shown in Figure 4.18. At every parent, a single packet with a variable data load is created from the children's data and transmitted up the tree. While data size increases, header size remains constant and the number of packets is reduced, reducing the total number of header bytes transmitted. This is equivalent to TAG [20] with atomic data.

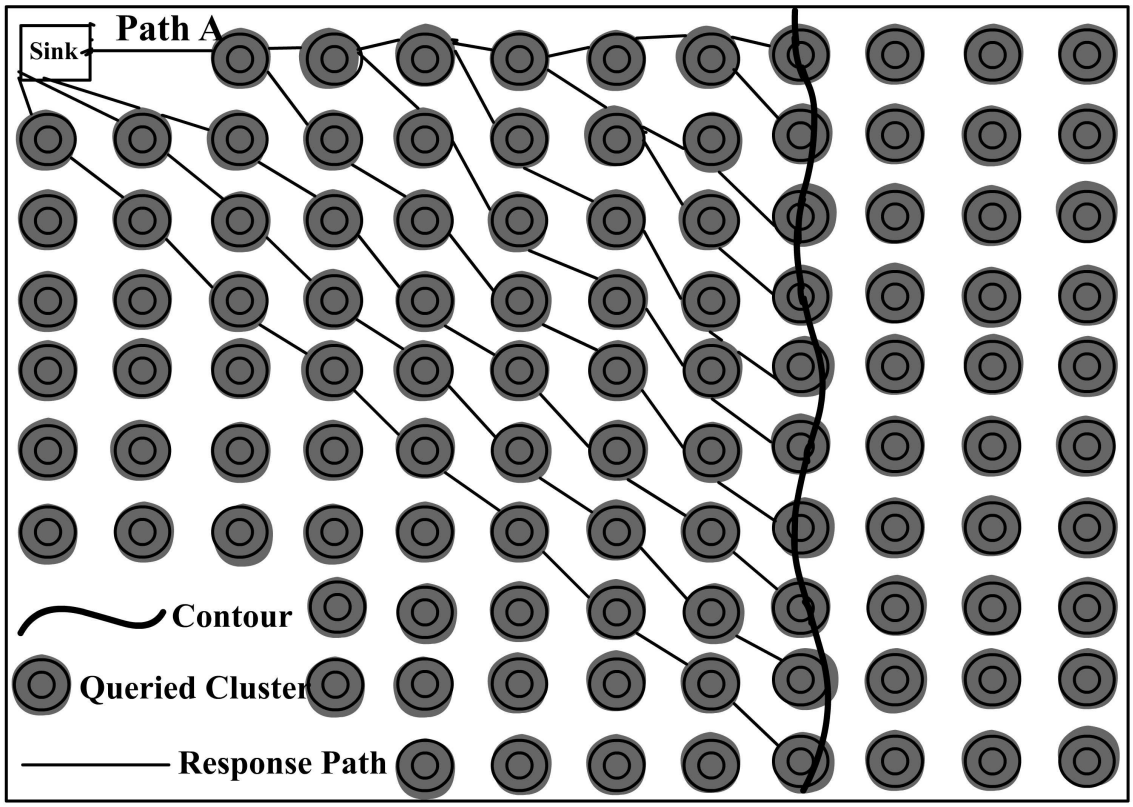

Figure 4.18: Illustration of the F/ATR algorithm

One advantage of this algorithm is that distributed in-network processing can be applied to the response data during propagation to the sink. Even though contour data is aggregated and transmitted to the sink in the shortest path there are possible

shortcomings. Consider the different paths in Figure 4.18, there are considerable savings of data along path A due to in-network aggregation of different packets while there are no savings along remaining paths which are similar to the general routing. The degree of savings along a particular route depends on the distribution of data in the network. Non-uniform data distributions result in suboptimal aggregation savings. This algorithm, like the F/SPR, requires the query to be propagated through the entire network which is costly and the response propagation to the sink is not efficient. To counteract these problems several novel tree-based routing algorithms are proposed in the later sections. A detailed description of the F/ATR algorithm is provided in Figure 4.19.

1. propagate the query request through the network as explained in the flooding query propagation technique

2. **CH/CM/GN:**
3. **On** receiving **query request:**
4. start an aggregation sink query response timer based on the node's hop count to the sink and wait for the query responses from the lower level nodes
5. process the received query request

6. **On** receiving **sink query response:**
7. **if** query is processed **then**
8. discard the received response
9. **else**
10. store the received query response
11. **end if**

12. **On aggregation sink query response** timer expiry**:**
13. **if** responses are present **then**
14. aggregate the received responses
15. forward the aggregated sink query response to the higher level nodes
16. **end if**
17. set the query processed to true

Figure 4.19: F/ATR algorithm

4.5.3 Information-driven Shortest Path Routing

I/SPR algorithm follows the contour and performs shortest path routing of the overall aggregated contour data present at each of the CHs to the sink independently. The current implementation uses the pattern based contour detection and cluster-based query propagation approaches to forward the query request along the contour and uses the shortest path routing technique to return the contour data to the sink as shown in Figure 4.20. This algorithm is included to help determine the relative impact of the query propagation and contour routing techniques. Overall, the query propagation through the network is efficient as the query is not flooded. However, routing the contour data to the sink is costly because the algorithm uses independent paths.

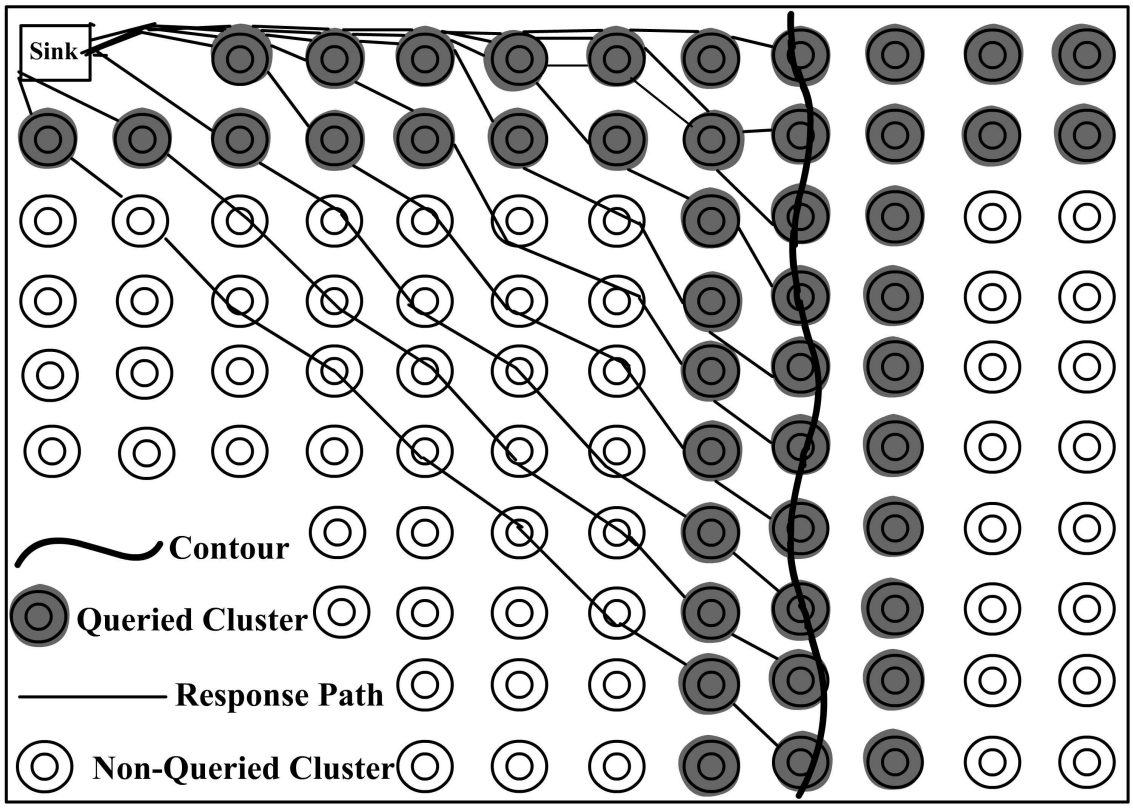

Figure 4.20: Illustration of the I/SPR algorithm

4.5.4 Information-driven Aggregation Tree-based Routing

I/ATR algorithm combines contour following to propagate the query and performs tree-based routing to forward the contour data to the sink. The only difference between F/ATR and I/ATR is the query is not flooded through the entire network using the current approach. Like in I/SPR algorithm, I/ATR algorithm uses pattern based contour detection and a cluster-based query propagation approach to forward the query request along the contour. On receiving the request, the CH waits for the responses from its members. If the members send their responses to the CH, indicating the presence of a contour, then the CH starts the synchronization timer and waits for synchronization with the other clusters. The synchronization timer is set sufficiently high to allow the query to propagate around the contour. The synchronization timer count is sent in the query request by the sink and is based on the network size, approximate number of network clusters or a priori knowledge of the contour. After synchronization, the aggregation timer at the CHs that detected a contour are set and the responses are propagated to higher layers using cascading timers as shown in Figure 4.21. The aggregation timer at the CH is dependent on the CH's hop count to the sink and the CH's hop count to the farthest node in the network. Intermediate nodes start their aggregation timers only after they receive the responses from lower layers. The value of the aggregation timer set at the intermediate nodes is dependent only on the node's hop count to the sink.

Overall, request and response propagations are efficient using this algorithm because the query request is propagated along the contour and the tree-based routing is used to aggregate the query responses and strip the unwanted response headers as the data is propagated along the tree to the destination. A detailed description of the I/ATR algorithm is explained in Figure 4.22. However, there is one major drawback to the aggregation tree for contour-based applications. Some responses can take different paths up the tree to the sink and may not get aggregated early enough to generate significant savings, which may result in individual headers getting forwarded inefficiently. However, this inefficiency can be avoided by exploiting the expected smoothness of the contour.

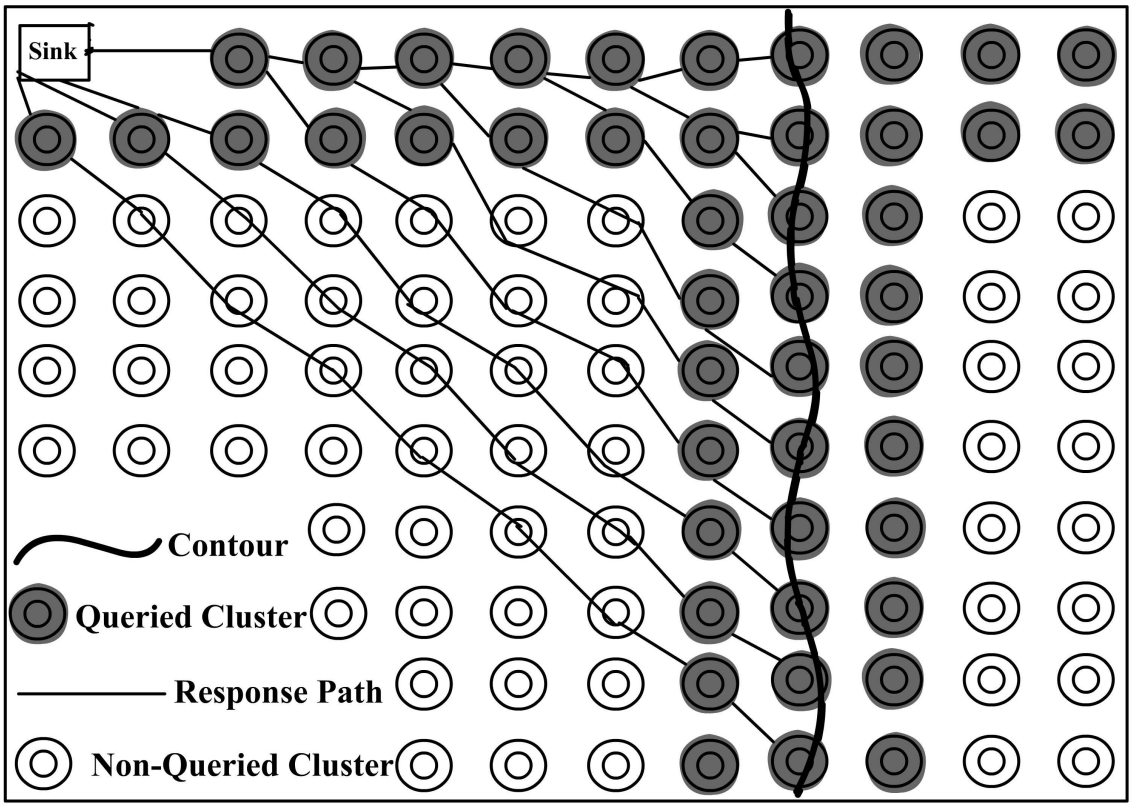

Figure 4.21: Illustration of the I/ATR algorithm

1. propagate the query request through the network as explained in the pattern-based contour detection and cluster-based query propagation techniques

2. **CH:**
3. **On member query response** timer expiry**:**
4. **if** member query responses are present **then**
5. start the synchronization timer at the CH
6. **end if**

7. **On synchronization** timer expiry**:**
8. start the aggregation sink query response timer expiry

9. **CH/CM/GN:**
10. **On** receiving **sink query response:**
11. **if** query is processed **then**
12. discard the received response

```
13.   else
14.       if aggregation sink query response is not started then
15.           start the aggregation sink query response timer
16.       end if
17.       store the received query response
18.   end if

19. On aggregation sink query response timer expiry:
20.   if responses are present then
21.       aggregate the received responses
22.       forward the aggregated sink query response to the higher level nodes
23.   end if
24.   set the query processed to true
```

Figure 4.22: I/ATR algorithm

4.5.5 Information-driven Contour/Aggregation Tree-based Routing

The I/CATR algorithm propagates the query along the contour and performs routing along the contour when necessary along with the tree-based routing to forward the contour data to the sink. Contours are natural phenomena which are generally smooth and continuous. Under normal network operation, if a cluster detects the contour, there is high chance that the neighbouring clusters have also detected the contour. This phenomenon is exploited by this algorithm. On detecting a contour, each CH checks if it is feasible to forward the overall aggregated response using the aggregation tree or the neighbouring CH that also has detected the contour as shown in Figure 4.23. In the I/CATR algorithm, the decision of whether to forward to the neighbouring CH or along the tree is made based on the hop distance to the sink. This decision is made during the cluster-based query propagation along the contour. On detecting a contour, the CH forwards the query to the neighbouring clusters in its vicinity to track the contour further and forward the query. The CH that initially detects the contour makes a decision to forward the response along the aggregation tree, as it has no information of its neighbouring clusters. However, this decision can be changed at a later point and a new path can be re-calculated once the neighbouring clusters detect the contour. Neighbouring CHs that have detected a contour check if it is shorter to forward their responses along the aggregation or to the CH that

has forwarded the query request. Similarly, the neighbouring CHs that have detected the contour also calculate if it is shorter for the CH that has forwarded the query request to forward its response through them. If so, they indentify the CH that forwarded the query request to change its forwarding path. Forwarding the overall aggregated CH response to the neighbouring CH that has also detected the contour helps in aggregating the responses along the contour and avoids individual CHs from forwarding the responses along the tree using different paths to the sink because neighbours along the contour are guaranteed to have data for aggregation while siblings in the aggregation tree might not. This approach further improves the efficiency of response propagation. A detailed description of the I/CATR algorithm is explained in Figure 4.24.

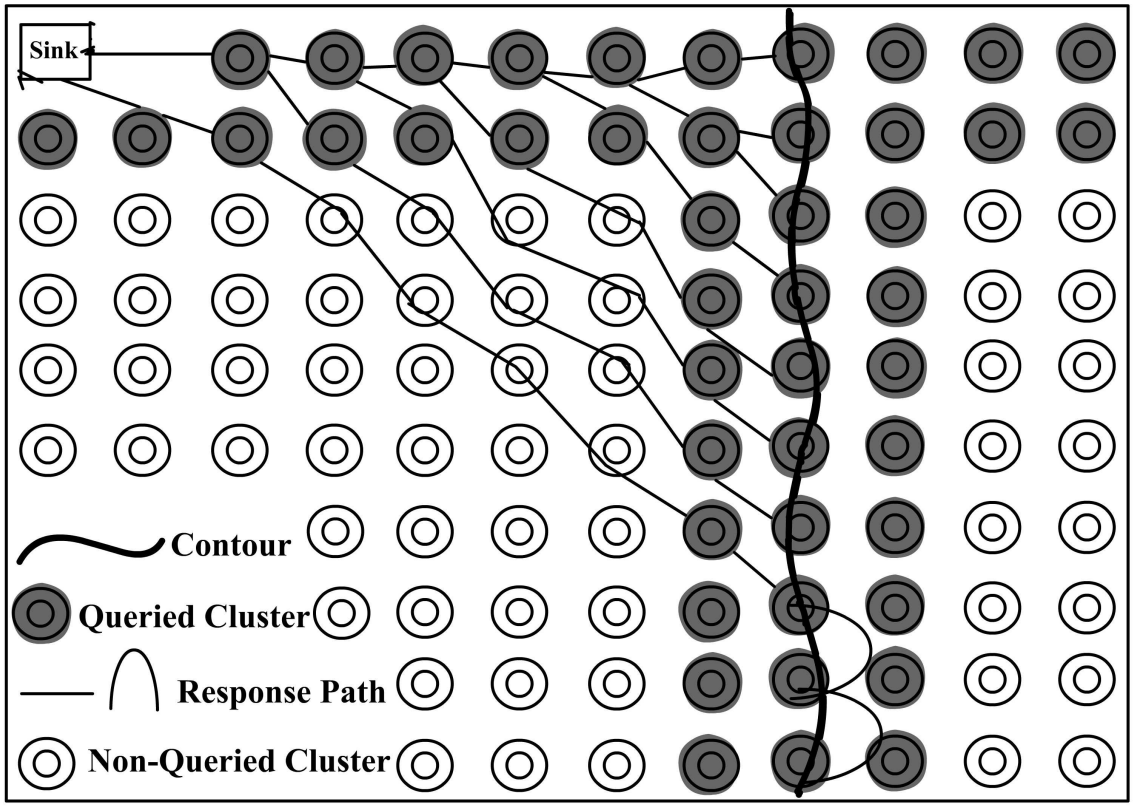

Figure 4.23: Illustration of the I/CATR algorithm

1. set the parent ID to the sink ID
2. propagate the query request through the network as explained in the pattern-based contour detection and cluster-based query propagation techniques

3. **CH:**
4. **On query response** timer expiry at the CH:
5. **if** member query responses are present **then**
6. **if** parent hop distance to the sink is less than hop distance of the parent to the sink through the current CH and parent is not the sink **then**
7. request the parent CH to route the response through the current CH
8. **end if**
9. **if** current CH hop distance to the sink is less than hop distance of the current CH to the sink through the parent CH **then**
10. set the parent ID to the current CH ID
11. **end if**
12. **for** each neighbouring CH which have also detected the contour **do**
13. **if** current CH hop distance to the sink through the neighbouring CH is less than hop distance of the parent to the sink **then**
14. set the parent ID to the neighbouring CH ID
15. **end if**
16. **end for**
17. start the synchronization timer at the CH
18. **end if**

19. **On synchronization** timer expiry**:**
20. start the aggregation sink query response timer expiry

21. **On** receiving **change parent ID request then**
22. **if** parent hop distance to the sink is greater than hop distance of the CH from which the request is received **then**
23. set the parent ID to the CH ID from which the request was received
24. **end if**

25. **CH/CM/GN:**
26. **On** receiving **sink query response:**
27. **if** query is processed **then**
28. discard the received response
29. **else**
30. **if** aggregation sink query response is not started **then**
31. start the aggregation sink query response timer
32. **end if**
33. store the received query response
34. **end if**

```
35. On aggregation sink query response timer expiry:
36.     if responses are present then
37.         aggregate the received responses
38.         if parent ID is same as the current CH ID or sink ID then
39.             forward the aggregated sink query response to the sink
40.         else
41.             forward the aggregated sink query response to the parent
42.         end if
43.     end if
44.     set the query processed to true
```

Figure 4.24: I/CATR algorithm

4.5.6 Information-driven Contour and Shortest Path Routing

I/CSPR algorithm is completely different from the contour data routing approaches discussed in previous sections because it assumes that the payload size is constant. In this algorithm, both the query and response are routed together. This approach is useful for aggregate queries such as MAX, MIN, AVG and SUM or for combined queries where a single parameter is required along a different contour (max pressure at T = 0). For example, consider a contour-based application that wants to find the maximum height on a particular temperature contour in a mountainous area. In this case, routing the response along the contour is feasible even if moving away from the sink as the contour data only contains the maximum height on the contour. Because the data size is constant there is little difference between propagating the query and the response. It also prevents clusters from forwarding individual responses up the aggregation tree and aggregating the contour data while being propagated which can be costly. However, this approach is not feasible if the accumulation of data size outweighs the header propagation savings of exclusive contour routing.

In this approach, the request is propagated along the contour using the cluster-based query approach with a slight modification. The request is propagated reliably along the contour with the response piggy-backed to it. On receiving the request, the neighbouring

CHs compare the received piggy-backed response with the member responses received and propagate the request further until the request can't be propagated any more as shown in Figure 4.25. Finally, CHs that couldn't further propagate the requests forward the responses to the sink using the one of the routing techniques proposed in the previous sections. This algorithm is not implemented in section 6 and is a logical extension presented for the completeness of the design and is part of the future work.

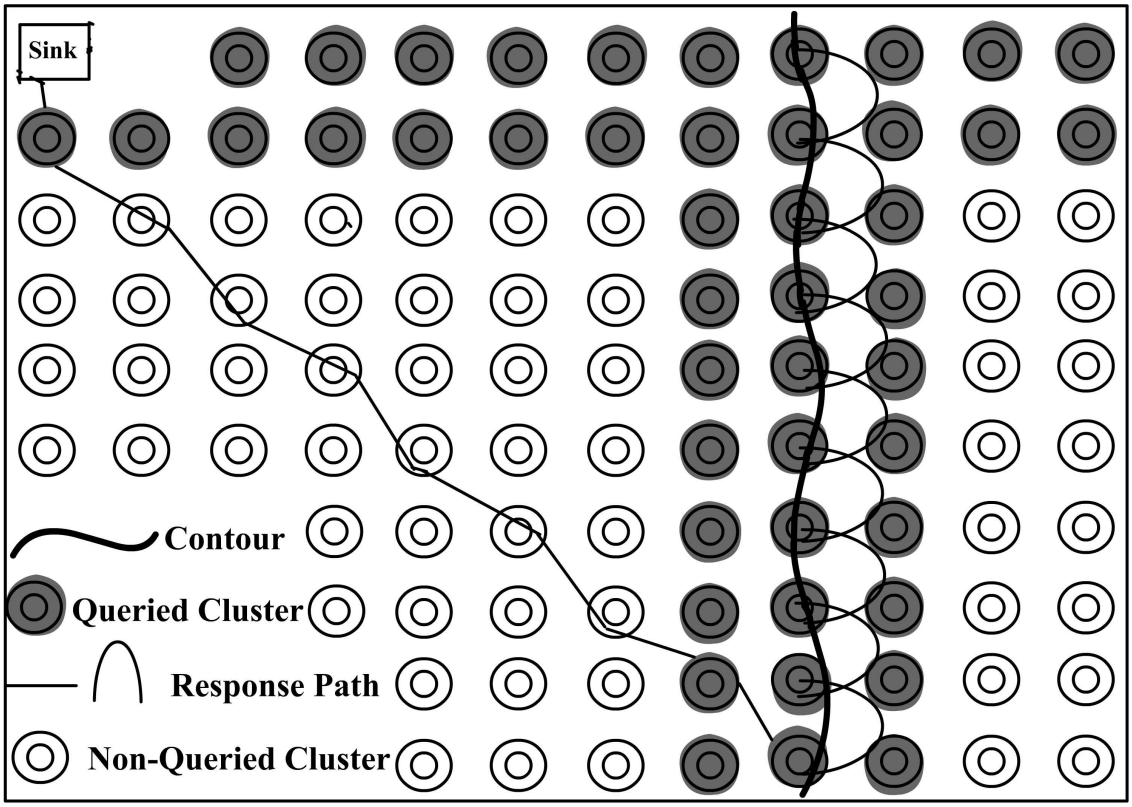

Figure 4.25: Illustration of the I/CSPR algorithm

4.6 Summary

F/SPR and F/ATR algorithms are the baseline shortest path and aggregation tree-based algorithms described in this thesis for comparison. The current algorithms in the literature that use energy efficient query propagation schemes either route the data to the sink by

aggregating the data in the reverse path or in the shortest path after aggregating along the phenomenon of interest. Moreover, routing each node's contour data independently to the sink using the shortest path is expensive. None of the existing algorithms provide a reliable and energy-efficient solution for contour-based applications. I/SPR, I/ATR, I/CATR and I/CSPR are novel algorithms proposed in this thesis. All these algorithms use the proposed reliable and efficient pattern-based query detection schemes and cluster-based query propagation schemes to forward the query in the network along the contour. The proposed cluster-based spatial and temporal suppression techniques can be used with these algorithms to remove redundant data.

The I/SPR algorithm routes the overall aggregated response at each CH to the sink along the shortest path independently. Though this is not efficient compared to the other proposed schemes, it is better than forwarding individual node data to the sink. The I/ATR algorithm uses a tree-based routing technique to route the data efficiently to the sink by performing in-network processing at the intermediate nodes. However, some of the responses can take different paths up the tree to the sink and may not get aggregated early enough to generate significant savings. The I/CATR algorithm provides guaranteed aggregation by performing routing along the contour when necessary along with the tree-based routing to forward the contour data to the sink.

CHAPTER 5

EXPERIMENTAL METHODOLOGY

In this chapter, the experimental setup for the simulations is described. To study the performance and scalability of the proposed algorithms, experiments are simulated using the NS-2 network simulator because physical experiments of sufficient scale performed are expensive, are difficult to control, are time consuming to setup and maintain and produce results that may be difficult to interpret.

5.1 System Modeling Assumptions

Topology

A grid topology is employed in the simulation as a grid topology is efficient in optimizing the sensor nodes placement, where the nodes are evenly placed based on their transmission and sensing range. The grid topology allows simple generation of routing tables, allowing us to test the proposed algorithms without having to implement clustering algorithms. Topological side-effects on the results are easier to detect in structured topologies, making conclusions more robust. Few truly random static deployments exist, making grid-like networks the norm for the problem that is being addressed in this thesis. The network is partitioned into two-hop clusters as shown in Figure 1.1. Each cluster consists of 17 nodes. Out of these, one node is a CH, eight nodes are CMs and the remaining nodes are GNs. The network distance between any two CHs is five hops. Nodes are densely deployed between the CHs. The reason for the dense deployment is to achieve greater detail in contour measurement and to permit efficient contour data suppression. CHs consume more power compared to the other nodes and can transmit their messages to their one and two-hop neighbours, whereas the CMs/GNs transmit messages only to their one-hop neighbours. CHs/GNs/CMs receive messages

only from their one-hop neighbours. CMs and CHs can transmit messages only within the cluster. GNs can transmit messages from within the cluster and to neighbouring clusters. A single sink is simulated.

Location Information

In many sensor network applications, knowing the spatial location of each node is required. Consider a contour application where the sink has requested the location of the 10 degree contour line. Without any information on the location of the nodes it is impossible to reconstruct the contour map without transmitting each nodes location individually. In the simulations, nodes don't transmit their locations to the sink because it is assumed that the sink knows the location of the nodes present in the network. This is a reasonable assumption for a static network.

Routing Tables

In sensor networks, the destination node may be one or more hops from the source node. In a multi-hop network, nodes need to forward the packets to their intermediate neighbours which in turn decide to forward the packet until the packet reaches the destination. For the nodes to forward the packets in an efficient manner, routing tables are used. In these simulations it is assumed that the nodes configure their routing tables statically before the query is propagated by the sink. Every node in the network knows the shortest route to the sink from itself. In addition, each node, based on its cluster membership, has additional routing entries to perform intra-cluster and inter-cluster routing. Again, these are reasonable assumptions for a static network.

5.2 Simulation Platform

NS-2, a discrete event network simulator used for modeling protocols for wired and wireless networks [45]. The simulator processes events until there are no events pending. NS-2 has a single thread of control, so there are no race conditions or deadlocks. The NS-2 architecture follows an object-oriented approach and is written in C++ and OTCL (Object variant of TCL). Most of the properties like reusability, abstraction, encapsulation and inheritance are supported. The core data processing is implemented in C++ which forms the back-end of the simulator. OTCL is used in the front-end to configure the simulation parameters, trigger actions in a periodic or event-based manner, and manipulate C++ objects, allowing changes to the parameters without re-compiling. The TCLCL library acts as an interface between C++ and TCLCL allowing them to share variables and functions. Both TCLCL and C++ share a class hierarchy. Packet and event tracing mechanisms are possible in NS-2. The packet tracing mechanism allows tracking of the transmitted, received or dropped packets on all links. The event tracing mechanism helps trace all the events that are triggered during the operation.

The Wireless model in NS-2 consists of the mobile node at the core, with additional supporting features such as the ability to transmit and receive signals to and from the wireless channels that allows simulations of multi-hop ad-hoc networks, wireless LANs and wireless sensor networks. Our proposed query propagation techniques, suppression algorithms and routing protocols are implemented on top of the mobile node protocol stack and makes use of the service provided by the stack.

5.3 Implementation of System Model

Topology Generation

The topology generator written in C creates a uniform cluster-based topology. The generator takes the length and breadth of the network as parameters and places the nodes uniformly as shown in Figure 1.1. The topology can easily be scaled by changing the length and breadth of the network. By default, the sink is placed at the top left most corner of the network. However, the sink can be relocated to a different location in the network by specifying its location in the topology generator. Each node's distance to the sink is also computed by the topology generator which is useful for configuring the routes to the sink.

Location Information Generation

Location information is generated from the topology automatically. Each point in the generated topology is represented by an X and Y coordinate. These generated node locations are used by the TCL script in configuring the mobile nodes in NS-2. Though the nodes don't transmit their location information in the response packets to the sink, the node location can be obtained based on their node ID.

Routing Entries Generation

Routing entries for the entire network are also generated based on the topology automatically. Every node in the network configures its own routing table from the generated routing entries. Each configured routing table consists of multiple routing entries. Some of these fields in these routing entries are optional and may be set or unset

depending on the node membership in the cluster. The following are the fields in a generated routing entry:

- Source node ID is the ID of the node itself
- Destination node ID is the ID of the destination
- Next hop node ID is the forwarding node ID
- GN ID is used by CHs to transmit the query request to neighbouring clusters
- Destination hop count is the distance to the destination in hops
- CH ID is the CH of the node's cluster
- Node membership indicates the role of the node within the cluster
- Node location indicates the position of the node

For all CMs/GNs/CHs, the first routing entry in the routing table is its route to the sink. The node location in this entry indicates the position of the node. The remaining routing entries in the table are either the routes to their CH or neighbours. In these routing entries, the node location indicates the destination node's location.

Scalar Field Generation

A scalar field generator is used for generating different continuous scalar fields. These fields are generated using randomly generated parameters about user provided set points and are dependant on the topology size. Figure 5.1 shows a generated continuous scalar field for an 800x800 meters network topology. Similarly, Figure 5.2 shows an equivalent contour map which is reconstructed at the sink from the received network contour data for the generated scalar field. Each point in the network has a specific value assigned to it. The continuous scalar field values generated are stored in a file. Based on a node's location in the network, a value is assigned. These values, once assigned, don't change in the course of the experiment. The generator is implemented in MATLAB and uses sinusoidal functions to generate the continuous field values. The generator takes the

network size, number of frequencies and frequency scaling as the input parameters. The network size is comprised of the network length and breadth. The number of frequencies indicates different random sinusoidal waves that are to be generated and frequency scaling determines the scaling of these waves. Random amplitude, phase, frequency and offset are generated for each wave from the input parameters. Finally, these waves are summed up as shown in the equation which is used to generate the values of the scalar field.

$$\text{Field} = \sum_{i=0}^{n} \text{A(i)} \times \text{Sin}\ (\text{F}x\ (\text{i})/\text{F}scale + \text{P}x(\text{i})) \times \text{Sin}\ (\text{F}y\ (\text{i})/\text{F}scale + \text{P}y(\text{i}))\ +\ \text{O(i)}$$

where n is the number of waves; A is the amplitude; F is the frequency;
P is the phase and O is the offset.

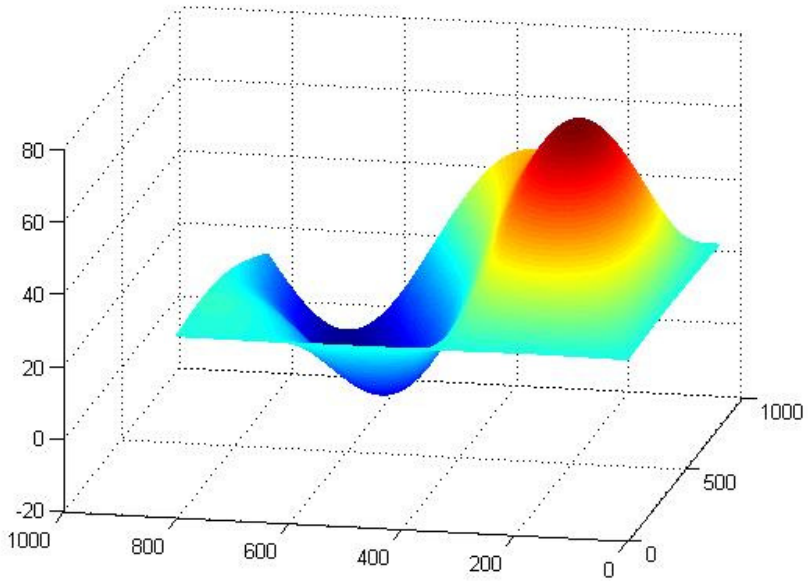

Figure 5.1: Continuous scalar field

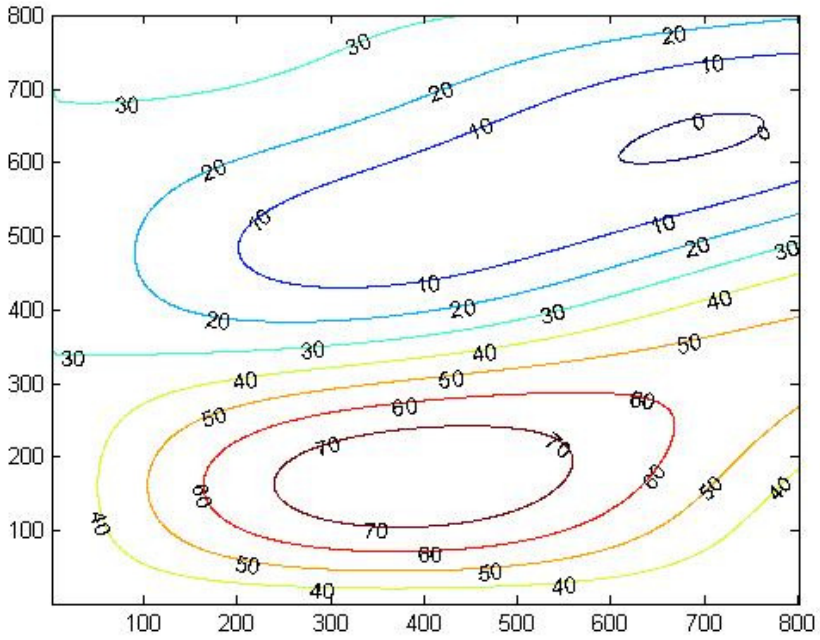

Figure 5.2: Contour Map

5.4 Simulation Execution

The algorithms in Table 4.1 are implemented in NS-2. Configurations such as the mobile node's protocol stack initialization, duration of the simulation, node's power settings, and node's location information are specified in the TCL script file, which triggers the start of the simulation.

Consider an example of detecting a 10 degree contour in an 800x800 meters network. Before the sink can broadcast the request into the network the simulation test-bed needs to be set up. First, the topology generator is configured with the 800x800 meters network setting. On running the generator, it internally generates a clustered grid topology as mentioned in the earlier sections. From this generated topology, node location information and routing files are created. Next, an 800x800 meters scalar field is generated. Each node's location information file is loaded by the TCL script files to

configure the mobile nodes. Later, when each of the mobile nodes is initialized the routing file is read and the routing table is set up based on the node ID.

The scalar field is used by the node to set its value based on the node location only after the query request is received from the sink. Nodes detect the presence of a contour between them on exchanging these values. To check the scalability of the algorithms various simulation experiments can be performed by varying the topology size. Similarly, to check the influence of contours on the algorithms experiments can also be performed by varying the shapes and sizes of the scalar fields.

CHAPTER 6

SIMULATION EXPERIMENTS

This chapter presents the results from the experiments that were performed. These experiments concern a wide range of issues such as the scalability of algorithms with network size, the impact of sink location on routing, the effect of suppression on traffic volume, query propagation through the network, and contour map reconstruction at the sink. In these experiments, the F/SPR, F/ATR, I/SPR, I/ATR and I/CATR algorithms are compared and results are plotted.

6.1 Effect of Suppression

In-network processing of data is important in WSNs as most of the data generated is spatially correlated and possibly redundant. Transmission of this redundant data in WSNs results in wasted network resources. Suppression is an important in-network processing technique that removes redundant contour data. This experiment examines the effect of cluster-based spatial suppression. In order quantify the gain using suppression, two kinds of experiments are performed. In the first experiment, the F/SPR, F/ATR, I/SPR, I/ATR and I/CATR algorithms are compared with and without suppression for a fixed topology of 800x800 meters and a single scalar field. In the second experiment, the I/CATR algorithm is run over 10 different scalar fields. Each of these scalar fields is generated uniquely by varying the amplitude, frequency and phases of the sinusoidal functions as explained in section 5.3. Finally, the data and message transmissions are compared for each of the protocols with suppression enabled and disabled. A sink is placed at a constant location at the upper left corner throughout all the experiments.

For testing suppression, a query should be propagated in the network by the sink pertaining to the phenomenon. In this experiment, the sink is interested in detecting a 40

degree contour present in the network. The F/SPR and F/ATR algorithms use flooding to propagate the query within the network, whereas the I/SPR, I/ATR and I/CATR algorithms use a pattern-based contour detection and cluster-based query propagation. For cases with suppression enabled the local suppression threshold set to one unit, which dictates the spatial suppression as explained in section 4.4.1.1. Overall, protocols with suppression enabled show a significant data and message saving compared to protocols with suppression disabled, as shown in Figure 6.1 and Figure 6.2. In the first experiment, F/SPR has a reduction of 30.6% in data volume and 22% in messages transmitted with suppression enabled. Similarly, I/CATR has a reduction of 35.7% in data volume and 28% in messages transmitted, clearly indicating that suppression has a significant effect in removing redundant contour data irrespective of the algorithm being used.

The second experiment clearly shows that the performance of the I/CATR algorithm with suppression is far better than without suppression for different contour shapes and sizes, as shown in Figure 6.3 and Figure 6.4. This indicates that suppression performs well for the entire class of contours considered.

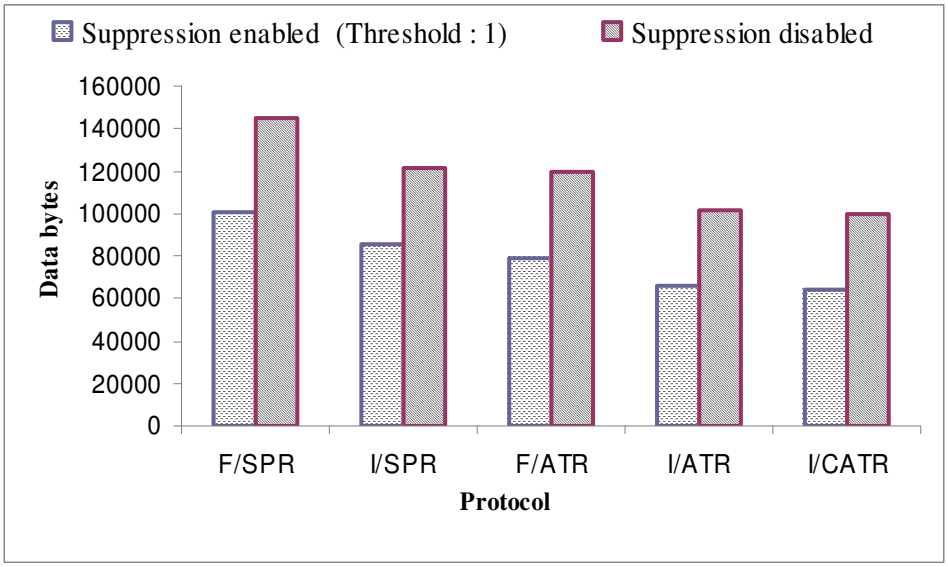

Figure 6.1: Effect of cluster-based spatial suppression on contour data

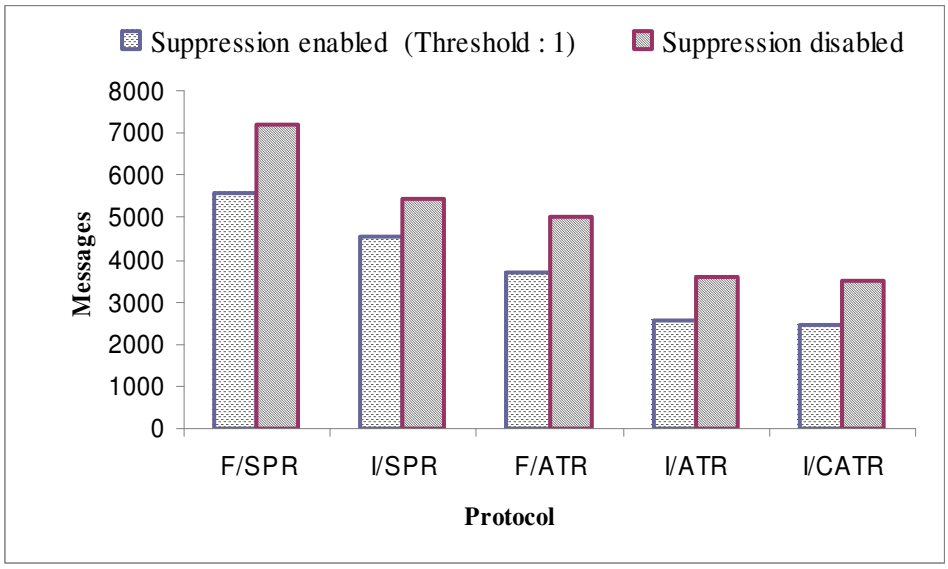

Figure 6.2: Effect of cluster-based spatial suppression on message transmissions

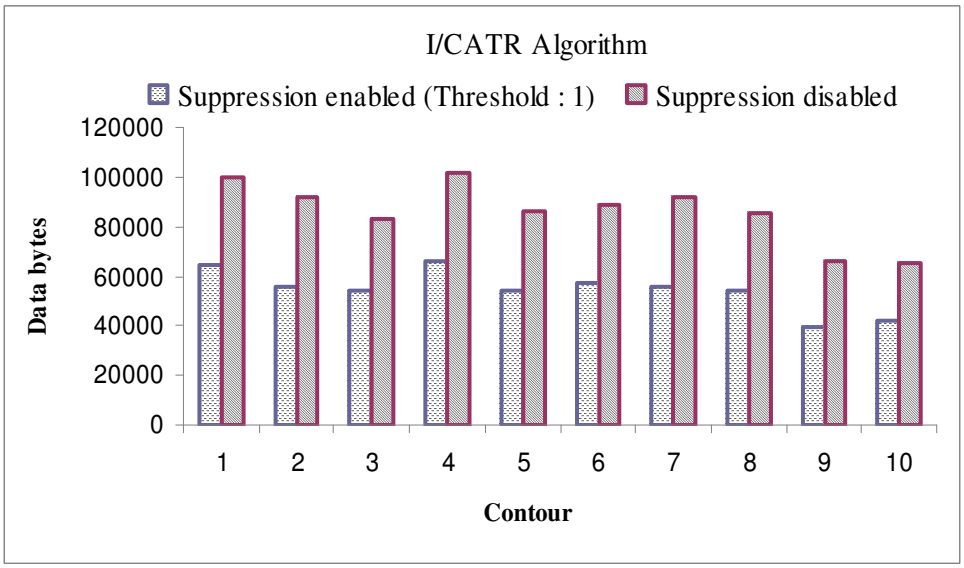

Figure 6.3: Effect of cluster-based spatial suppression on data for different contours

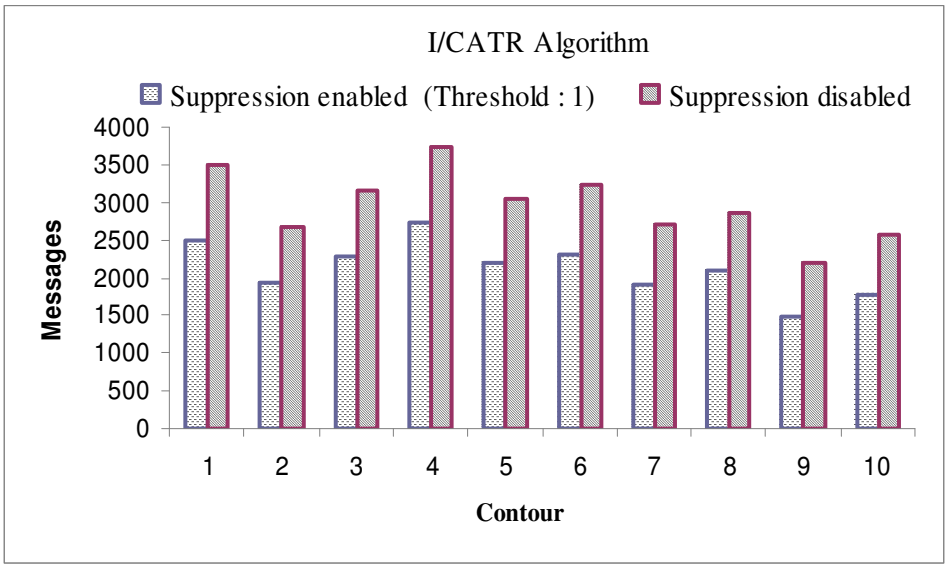

Figure 6.4: Effect of spatial suppression on messages for different contours

The primary reason for cluster-based suppression outperforming in all cases is because of the suppression of control messages and redundant data. In the two-hop clustered topology considered, only the CMs that detect the contour are allowed to broadcast their sensed readings along with the query response to the CH while the other CMs are restrained from broadcasting their sensed readings. Moreover, significant data savings are gained by performing three levels of suppression, one at each node type, to remove the redundant data. The GNs perform the suppression of their readings using the minimum contour distance suppression threshold parameter. Similarly, CMs perform the suppression of their readings using the minimum contour suppression threshold parameter and also from the received GNs and CMs query responses. Finally, the CH performs the suppression on all the received responses from the members. If members have no contour data to transmit after performing suppression, then they do not transmit any response messages, saving message transmissions.

6.2 Varying the Suppression Threshold

As explained in the previous section, suppression plays a vital role in removing redundant data and hence saving energy. The suppression is actually governed by the minimum contour distance suppression threshold parameter, which specifies the required spatial resolution required by the user. For example, a threshold of 5 units indicates that only points on the contour that are 5 meters apart are of interest, so the readings which are 5 meters apart are reported to the sink while the other readings are discarded. The suppression threshold gives the flexibility to tune the amount of suppression that is required. This threshold value is propagated to the nodes in the query. The goal of this experiment is to determine how the amount of redundant data removed depends on the suppression threshold value.

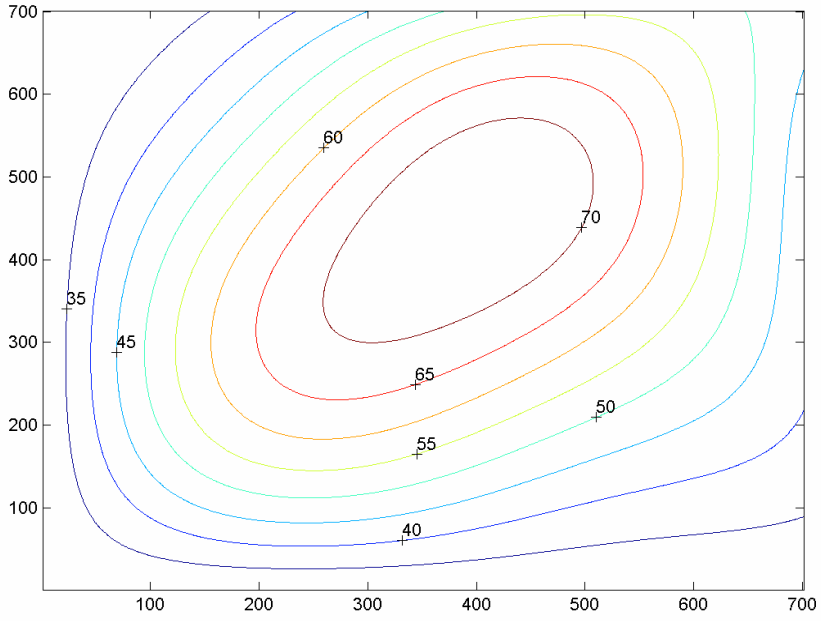

Figure 6.5: 700x700 meters contour map

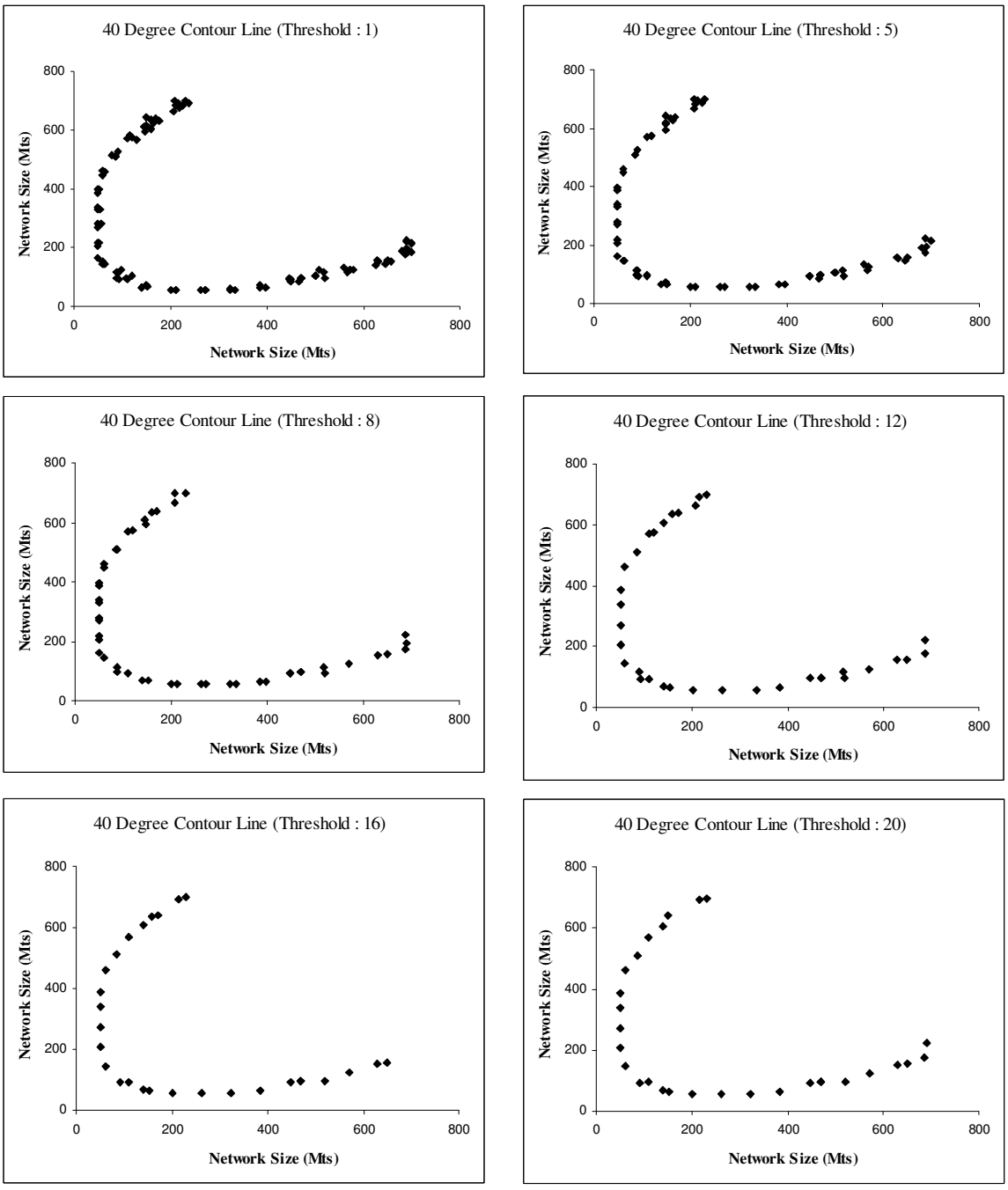

Figure 6.6: Contour reconstruction at different suppression thresholds

In order to verify that the removed redundant data is proportional to the suppression threshold, a fixed topology of 700x700 meters network is considered as shown in Figure 6.5. The I/ATR algorithm is run with 6 different suppression thresholds and a

single scalar field with cluster-based spatial suppression enabled, as shown in Figure 6.6**.** Query propagation is performed as explained in the previous sections. The results clearly show that the removal of redundant data is proportional to the suppression threshold, as shown in Figure 6.7 and Figure 6.8. The percentage reduction in data volume and message transmission by varying the suppression thresholds between 1 and 20 is 37.5% and 2.5% respectively. The significant data volume reduction is due to the efficient spatial suppression performed using the threshold value and the reason for marginal message gain is because nodes transmit their responses even if they have only a small amount of contour data present. This clearly shows that the efficiency of an algorithm in WSNs cannot be judged based solely on message transmissions, but should also consider the payload data transmitted in these messages. A closer look at the results shows that as the suppression threshold value increases, the marginal compression decreases.

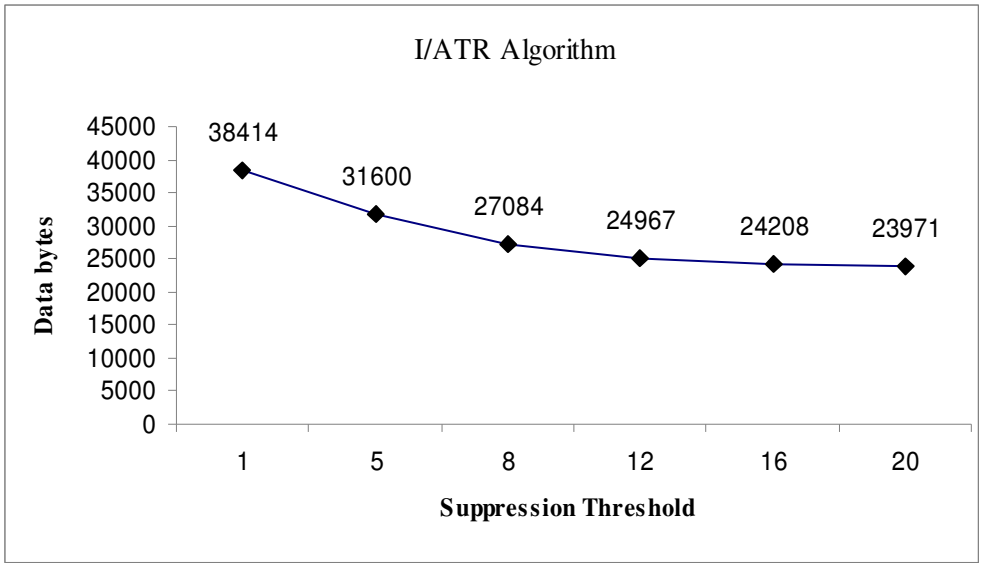

Figure 6.7: Effect of suppression threshold on suppression of contour data

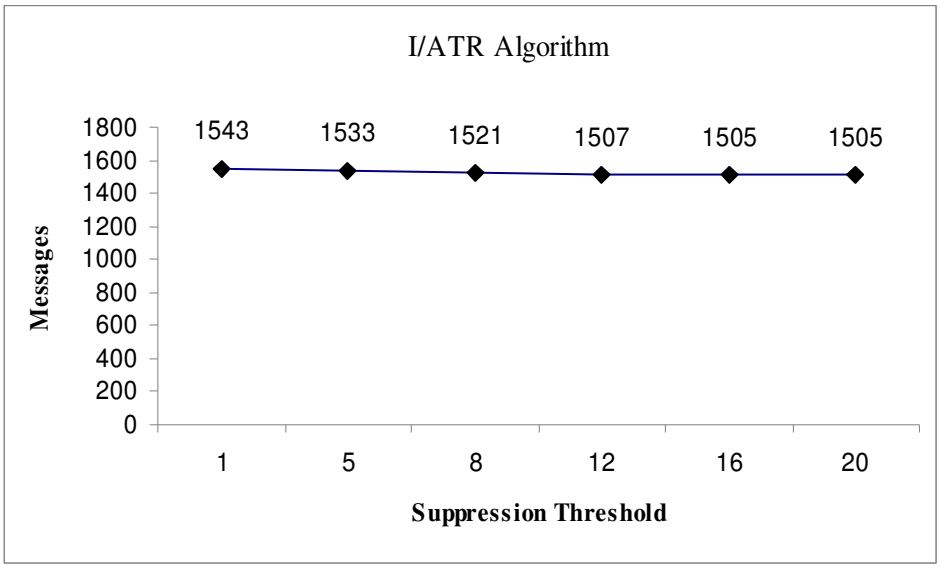

Figure 6.8: Effect of suppression threshold on suppression of message transmissions

6.3 Impact of Sink Location

The sink can reside anywhere in the network. Its location has an impact on the data volume and message transmissions for various protocols. The farther the sink is from the contour, the greater the data volume and message transmission required. The pattern-based contour detection and cluster-based query propagation are dependent on the distance from the contour from the sink. Similarly, the response transmission is also dependant on the sink location. To assess the impact of sink location on the F/SPR, F/ATR, I/SPR, I/ATR and I/CATR algorithms an 800x800 meters network is considered. Each protocol is run for 6 different sink locations and a single scalar field with cluster-based spatial suppression enabled and the suppression threshold set to one unit.

The I/CATR algorithm outperforms all the algorithms in terms of data and message transmissions irrespective of the sink locations, as shown in Figure 6.9 and Figure 6.10. When the sink is placed at 220x25 meters, the data volume reduction of the I/CATR algorithm is 42.7% more than with the F/SPR algorithm and 16.5% more than

with the I/SPR algorithm. Similarly, when compared to F/ATR and I/ATR algorithms, the data volume reduction is 26.4% and 3% respectively. Similarly, for a sink location at 35x355 meters, the data volume reduction of the I/CATR algorithm over F/SPR and I/SPR algorithms is 35.3% and 14.8% respectively. When compared to tree-based protocols (the F/ATR and I/ATR algorithms), the data volume reduction is 23.5% and 6.4% respectively. The results show that with two different sink locations, the data volume and message traffic varies depending on the proximity of the contour to the sink. However, the ordering of the performance of the protocols is independent of the sink location.

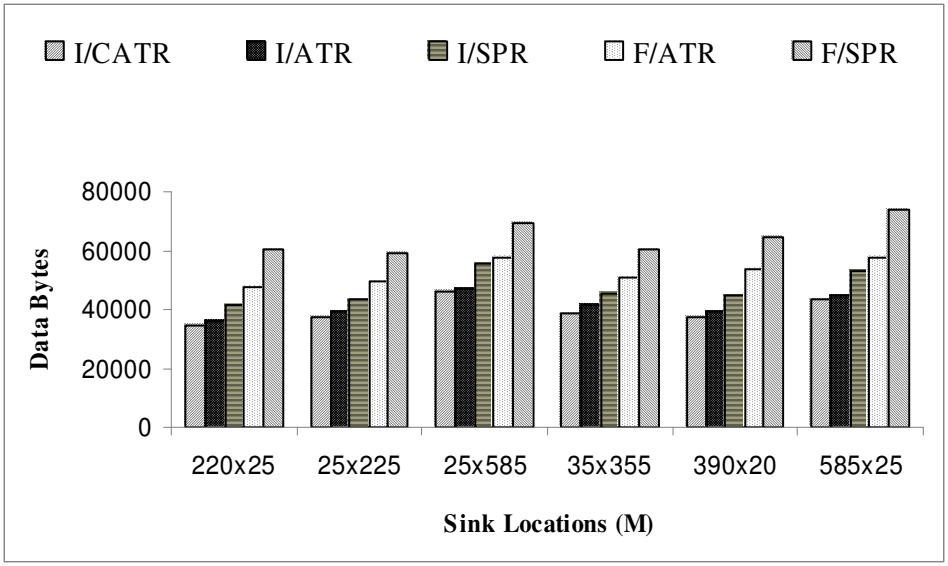

Figure 6.9: Impact of sink location on contour data

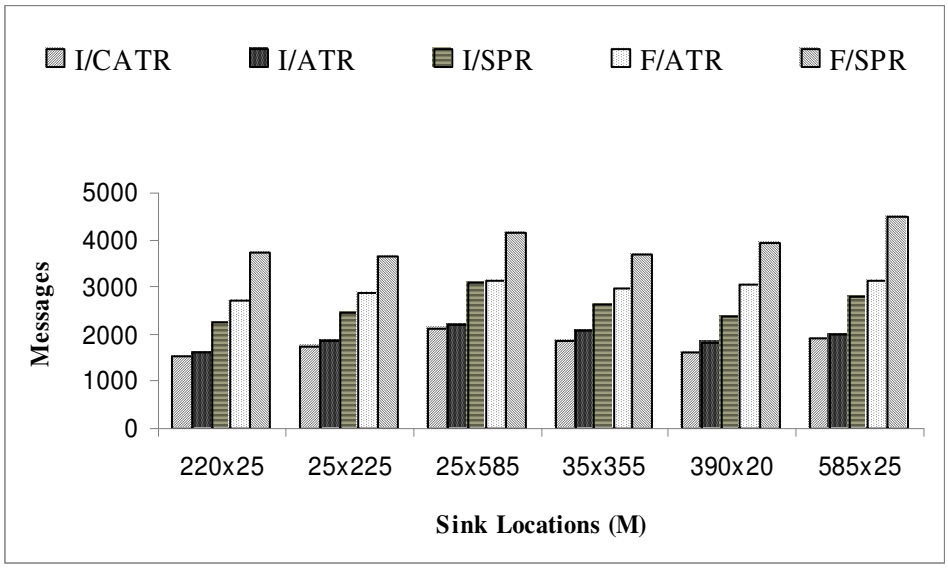

Figure 6.10: Impact of sink location on message transmissions

6.4 Contour Dependence

Contours vary in shape and size. For example, a contour which covers the entire network requires more data to represent and messages to transmit the data than a smaller contour. With flooding, the data and message transmissions are constant and depend on the size of the network. However, in the case of pattern-based contour detection and cluster-based query propagation the efficiency depends on the size and shape of the contour. Greater contour size results in the query being propagated greater distances. Normally, contours are smooth and it is unlikely for a contour to pass through all the clusters in large networks, and therefore it can be expected that pattern-based contour detection and cluster-based query propagation techniques seldom propagate the query to all the clusters in the network, unlike flooding. The goal of this experiment is to assess the dependence of performance on the shapes and sizes of the contours.

All the protocols were evaluated on a fixed topology of 800x800 meters over 10 different scalar fields with cluster-based spatial suppression enabled and suppression threshold set

to one unit. Finally, the percentage reduction in data volume and message transmissions is calculated for each of the contour algorithm with respect to I/CATR. On obtaining the percentage reduction, average, maximum and minimum reduction percentages are computed for each of the protocols, as shown in Figure 6.11 and Figure 6.12.

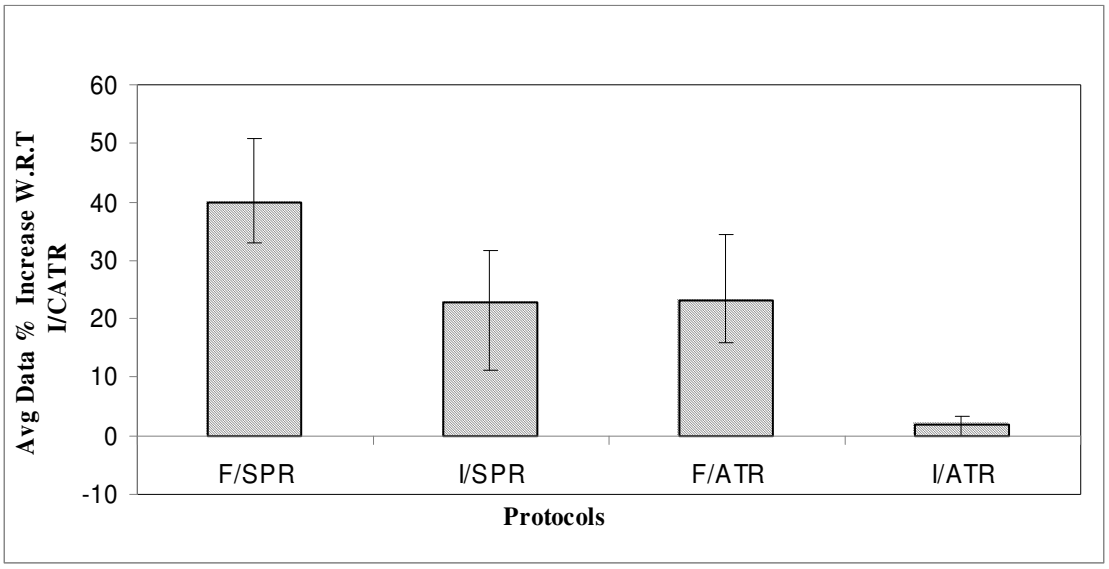

Figure 6.11: Influence of contour shapes and sizes on contour data

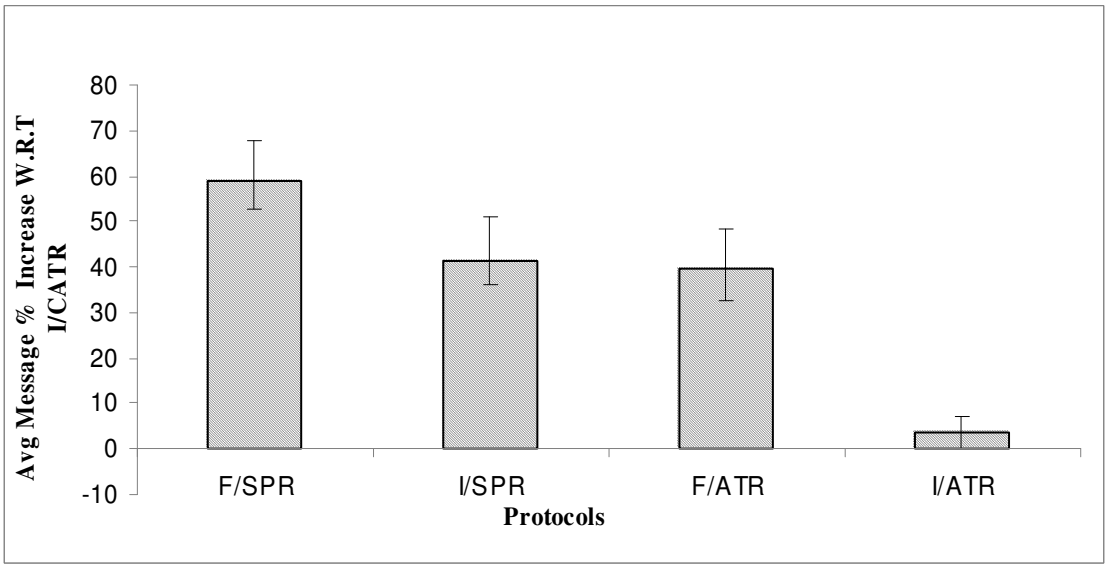

Figure 6.12: Influence of contour shapes and sizes on message transmissions

Minimum and maximum reduction percentages are represented by the error bars. Comparing the F/SPR and I/SPR algorithms with the baseline I/CATR algorithm, the average data volume reduction is 40% and 22.75%, maximum data volume reduction is 50.95% and 31.67% and minimum data volume reduction is 33% and 11.15% respectively. The average message transmission reduction is 59% and 41.4%, maximum message transmission reduction is 67.6% and 51% and minimum message transmission reduction is 52.7% and 35.9%.

In comparison to tree-based protocols, the I/ATR and F/ATR algorithms average data volume reduction is 1.81% and 23%, maximum data volume reduction is 3.37% and 34.34% and minimum data volume reduction is -0.13% and 16.11% respectively. The average message transmission reduction is 3.77% and 39.45%, maximum message transmission reduction is 7.3% and 48.5% and minimum message transmission reduction is -0.05% and 32.4%. The results clearly show that the data and message transmissions of the protocols are influenced by the contours shape and size. There are several reasons that explain this behavior with respect to query request propagation in the network and receiving the response regarding the phenomenon from the network. In case of query flooding, the query request transmissions are constant and vary depending on the network size. However, with the proposed energy-efficient contour detection and query propagation schemes the message transmissions to detect the contour and propagate the query request along the contour depends on the location, shape and size of the contour. Similarly, to route back the responses to the sink, the data and message transmissions depend on the contour. However, I/ATR and I/CATR algorithms outperform the other algorithms in terms of data and message transmissions irrespective of contour location, shape and size.

The primary reason for very high byte and message transmission counts in the F/SPR algorithm is it floods the network with the query, which is unnecessary for contour-based WSN applications. Moreover, the F/SPR algorithm forwards the data to the sink independently, resulting in extra packet header overhead, because every cluster transmits

its own packet. Though the F/ATR algorithm uses flooding to propagate the query like in the F/SPR algorithm it performs better than the F/SPR algorithm because the responses get aggregated as they move up the aggregation tree, reducing the packet header overhead. The I/SPR algorithm is still similar to the F/ATR because it transmits the responses to the sink like in the F/SPR algorithm even though it uses pattern-based contour detection and cluster-based query propagation. This response results in extra overhead which offsets the gain achieved through efficient query propagation. The I/ATR algorithm performs better than most of the algorithms because it uses efficient query techniques to detect and propagate along the contour and aggregation tree during the data response phase. However, if response packets take different routes up the aggregation tree, then aggregation may be less efficient than in I/CATR where aggregation happens opportunistically along the contour first.

The I/CATR algorithm aggregates CHs responses along the contour before the responses are forwarded up the aggregation tree, resulting in better performance compared to other protocols. However, the data response transmissions to the sink in both the I/ATR and I/CATR algorithms are similar if I/CATR finds that routing all the responses to the sink is feasible only through the aggregation tree rather than using the contour based routing. Under these conditions, the overhead in the I/CATR algorithm due to control signaling for setting up the parent ID is slightly higher than the I/ATR algorithm. This is clearly visible in the results where the minimum data volume reduction percentage of the I/ATR algorithm was -0.1% in comparison to the I/CATR algorithm. On an average across multiple cases, the I/CATR algorithm performs better as shown in this experiment.

6.5 Network Scalability

WSNs are deployed for monitoring various natural phenomena like temperature, pressure and humidity, and vary in size depending on the application. For example, a contour application may require a medium to large scale sensor deployment in order to track the

contours efficiently over larger areas. Therefore, protocol performance should be scalable with network size. Contours may be present away from the sink in large WSNs. In this case, query propagation from the sink and response transmissions to the sink depend on the network scale. The impact of query request and response packet overhead may not be apparent in smaller networks. The goal of this experiment is to assess how scalable the proposed algorithms are with respect to network size.

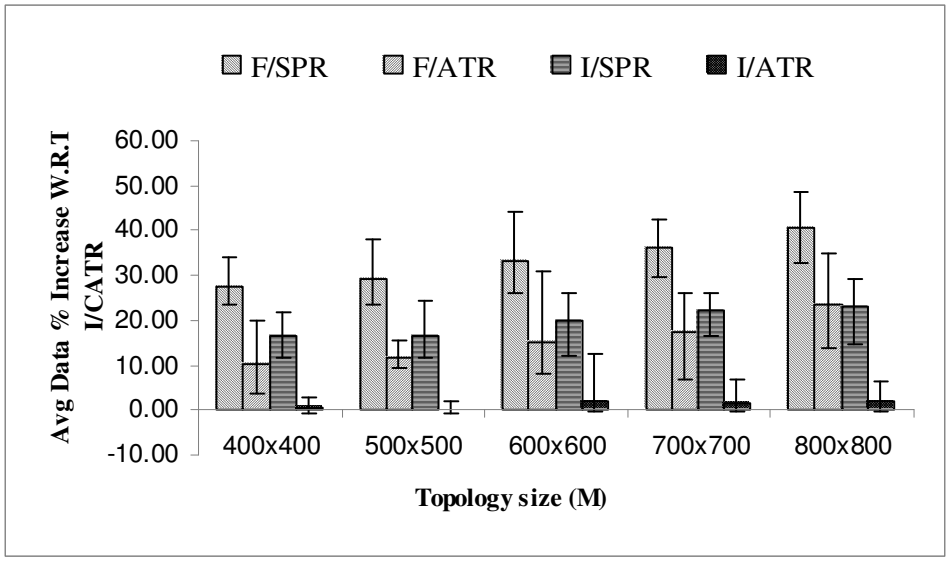

Figure 6.13: Effect of network scalability on contour data

Five different network topologies of 400x400 meters, 500x500 meters, 600x600 meters, 700x700 meters and 800x800 meters are considered. The number of nodes range from 792 to 3032. Ten different scalar fields are generated for each of the topologies for a total of 50 scalar fields. Each of the algorithms is run over all topologies and contours. Cluster-based spatial suppression is enabled during the experiment and the suppression threshold is set to one. Finally, the percentage reduction in terms of data volume and message count is calculated for each of the contours run under other protocols with respect to the I/CATR algorithm. On obtaining the reduction percentages, average, maximum and minimum reduction percentages are computed for each of these protocols

in terms of data and message transmissions, as shown in and Figure 6.14. Minimum and maximum reduction percentages are represented by the error bars.

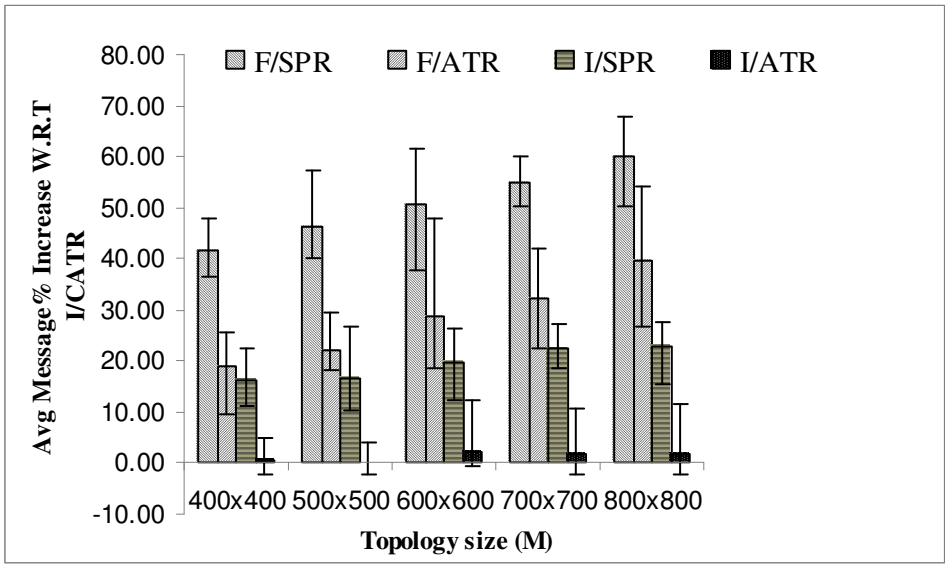

Figure 6.14: Effect of network scalability on message transmissions

Comparing the F/SPR and I/SPR algorithms with the baseline I/CATR algorithm for an 800x800 meters network the average data volume reduction is 40.8% and 22.8%, maximum data volume reduction is 48.5% and 29% and minimum data volume reduction is 32.5% and 14.6% respectively. The average message transmission reduction is 59.8% and 43.1%, maximum message transmission reduction is 67.9% and 48% and minimum message transmission reduction is 50.4% and 35.7% over the F/SPR and I/SPR algorithms. In comparison to tree-based protocols for a 800x800 meters network, for the I/ATR and F/ATR algorithms the average data volume reduction is 1.7% and 23.6%, maximum data volume reduction is 6.4% and 35% and minimum data volume reduction is -0.1% and 13.7% respectively. The average message transmission reduction is 3.6% and 39.9%, maximum message transmission reduction is 13.3% and 54.2% and minimum message transmission reduction is -0.1% and 26.7%.

Overall, results clearly show that the I/CATR algorithm outperforms all the protocols in terms of data and message transmissions. The I/CATR algorithm performs better because it detects the contour and propagates the query request along the contour independent of the network size. Moreover, it uses efficient routing along the contour and the aggregation tree to route the responses to the sink by performing in-network processing. On the other hand, the F/SPR and F/ATR algorithms use flooding to propagate the query in the network resulting in the query request transmissions which vary depending on the network size. The F/SPR and I/SPR algorithms cannot perform in-network processing because these algorithms route the responses to the destination independently, resulting in an increase in response packet header overhead with an increase in network size. The F/ATR and I/ATR algorithms use the tree-based approach in routing the packets to the destination by performing in-network aggregation, but there are chances that the packets take different paths up the tree to the destination and might not get aggregated until the packet reaches higher levels in the tree. The levels at which the packets get aggregated might increase with the increase in network size.

6.6 Contour Reconstruction

In order to reduce resource wastage the redundant contour readings are removed using the proposed cluster-based spatial suppression techniques. Cluster-based suppression uses a spatial resolution specified by the sink to suppress the contour readings. Suppression of contour readings reduces the number of readings encoded in the response payload. However, removing the redundant contour readings using suppression is not sufficient, as there is still a chance of resource wastage if all the elements of the valid contour readings are encoded into the response as the payload size increases. To avoid this wastage of resources, efficient compression techniques are used to reduce the elements that are encoded into the payload for each of the contour readings. The elements that are encoded after compression are the node ID, relative neighbour node ID and node's proximity to the contour, as explained in section 4.4.1.1.

After performing in-network processing the suppressed data is routed efficiently to the sink. On receiving the response payload, the sink decodes the payload and reconstructs the contour. The goal of this experiment is to demonstrate that the contour reconstructed at the sink is similar to the generated contour under observation. In order to verify the reconstruction of the contour, a fixed topology of 800x800 meters network is considered. The I/CATR algorithm is run with 4 different scalar fields with cluster-based spatial suppression enabled and suppression threshold set to one unit. The sink queries for a contour value of 40 using the pattern-based contour detection and cluster-based query propagation. Finally, the contour points reconstructed from the decoded contour data received at the sink from the network are overlapped onto their respective scalar fields, as shown in Figure 6.15. This clearly shows that the reconstruction of a contour value of 40 at the sink from the received network data is accurate.

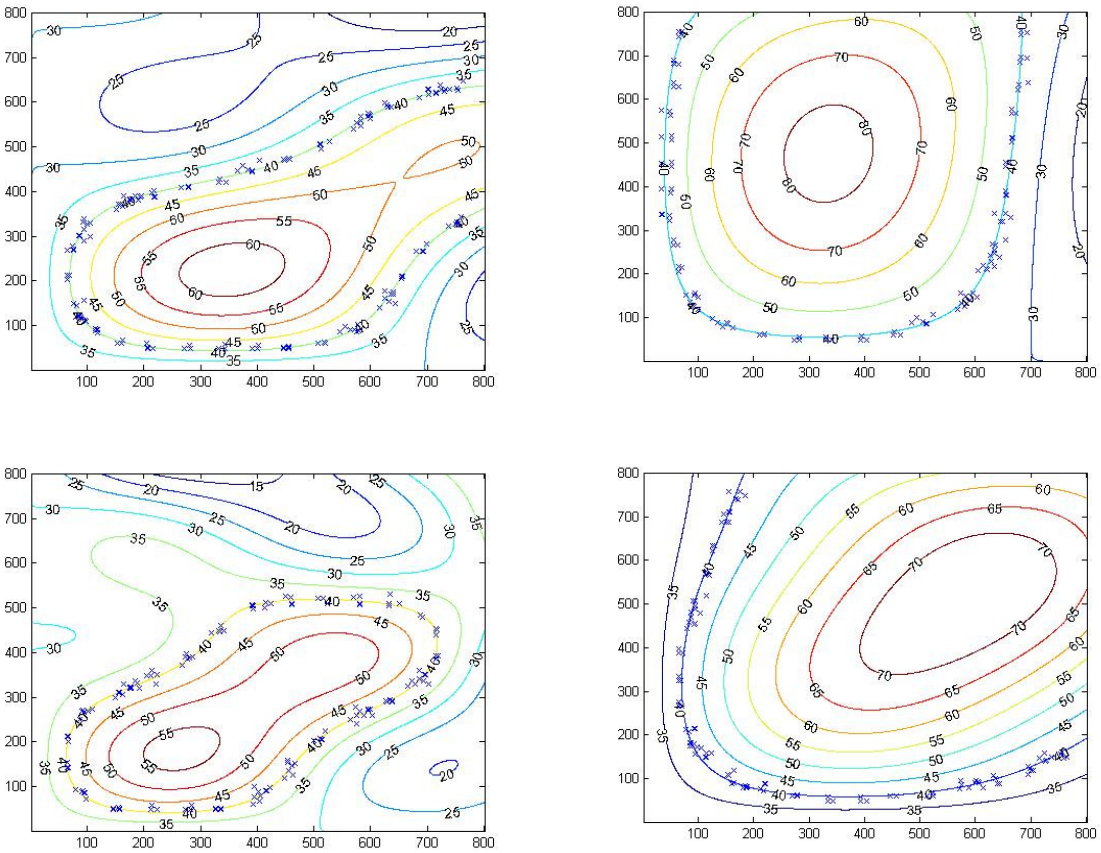

Figure 6.15: Reconstructed contour points ('x') are overlapped onto the scalar fields

6.7 Summary

The accuracy, dependence, performance and scalability of the F/SPR, F/ATR, I/SPR, I/ATR and I/CATR algorithms has been analyzed by performing various experiments, the results of which are reported in this chapter. The I/CATR algorithm is shown to be superior to all the other algorithms in most of the scenarios because it propagates the query efficiently along the contour independent of the network size, performs efficient cluster-based spatial suppression and routes the responses to the sink in an efficient manner along the contour or using an aggregation tree, depending on which is feasible, ensuring in-network processing at the intermediate nodes. The spatial suppression techniques used to suppress the spatially correlated data performed well compared to the no suppression paradigm for all the algorithms.

CHAPTER 7

CONCLUSIONS

Technology advancement in the recent years has enabled use of WSNs to monitor, detect and track external phenomenon with little human intervention. One type of WSN application, concerned with detecting and tracking contours is considered in this thesis. Contour-based WSN applications are applicable in many areas such as diagnosing network health, tracking moving vehicles, and tracking changing, spreading or diffusion of external phenomenon like temperature, pressure and humidity. These applications give an overview of the sensor field by constructing contour maps at the sink from the readings received from the network. WSNs are energy-constrained and to increase the network longevity the network resources should be used in an efficient manner while providing the desired results. This thesis has studied the problem of efficient design of contour-based applications in detail and provided an energy-efficient end-to-end network solution.

7.1 Thesis Summary

The focus in this thesis is contour-based WSN applications which give an overview of the network by constructing contour maps. A contour-based WSN application consists of three main components: contour detection and query propagation, in-network processing, and response routing to the destination. None of the current approaches have provided an overall solution for contour-based applications. Most of the existing research focuses on providing solutions for individual components and these solutions provided do not focus on co-existence with other components. Moreover, some solutions provided are generic for WSNs and may not feasible or efficient for contour-based applications. In this thesis, each of these components is examined in detail and an energy-efficient end-to-end

solution is provided. Some of the solutions and techniques provided can also be used with other WSN applications.

Contour Detection and Query Propagation

Detection and query propagation consider detecting the contour and propagation of the query along the contour in an efficient manner after detecting it. Some phenomenon being monitored, detected or tracked by a WSN may or may not be spread uniformly throughout the network. If the phenomenon is spread throughout uniformly, then flooding is the best option to propagate the query. However, if the phenomenon is not uniform, then flooding may not be a wise option. The localized phenomenon considered in this thesis is contours. To detect the contour a pattern-based contour detection algorithm has been proposed. Once the contour has been located an efficient and reliable cluster-based query propagation algorithm has been proposed for routing the query along the contour. Results clearly show that the contour detection and query propagation approaches are scalable and more efficient than query flooding. While this algorithm opens the door to several interesting subsequent inquires, these have been left to future work.

In-network Processing

Contour readings generated by the nodes in a WSN are often spatially or temporally correlated. Transmitting all these readings to the sink can waste network resources. Clever suppression of these readings using in-network processing helps prolong network life. In this thesis, two in-network processing schemes are proposed. One is a cluster-based spatial suppression scheme which suppresses the messages and data within the cluster without significant overhead. In this scheme, members belonging to the cluster perform efficient suppression based on a spatial resolution threshold. The other scheme is used along with suppression to reduce the actual data in the response payload using efficient encoding techniques. These techniques intelligently encode minimal data in the

payload for each valid contour reading, so that the sink reconstructs the contour with this information using interpolation. Results show that the savings due to cluster-based spatial suppression is dominant in terms of energy savings and the contour is reconstructed accurately at the sink.

Data Routing

Nodes that detect the contour need to route their query responses to the sink to reconstruct the contour map of the sensor field. Efficient routing of the query responses to the sink helps save network resources. In general, in WSNs, routing to the sink is performed by nodes independently using shortest path routing or aggregation trees. Routing of the contour data independently using the shortest paths to the sink misses the chance of stripping the response packet headers by consolidating the packets at the intermediate nodes. This results in unnecessary overhead and results in resource wastage. On the other hand, aggregation trees help in consolidation of headers, but the response packets may take different paths along the tree and might not get aggregated until the data reaches the higher levels in the tree due to the non-uniformity of contours. In some information-driven approaches where the query is routed based on the information gathered from the surroundings, the response data is either aggregated with the query and propagated or aggregated in the reverse path of the query. In these approaches, any hop which is not on the shortest path to the sink would be costly as the payload is usually much larger than the header in contour applications. This thesis proposed two classes of routing protocols for contour-based WSN applications based on shortest path and tree-based routing. I/SPR algorithm uses shortest path routing to forward the response to the sink. However, all the node's responses are aggregated at the CH before routing the overall aggregated response to the sink, reducing individual node transmission cost. I/ATR and I/CATR are tree-based routing algorithms. In the I/CATR algorithm, the routing is done along the contour or the aggregation tree, depending on which is locally more efficient. This approach tries to reduce the frequency with which response packets

from neighbouring clusters take different paths up the aggregation tree by aggregating along the contour instead.

Evaluation

Querying, in-network processing and data routing algorithms are evaluated using simulations and their performance is analyzed. The scalability of the proposed algorithms is evaluated by scaling the network size from 400x400 meters to 800x800 meters. For a 400x400 meters network of 792 nodes running the F/SPR and F/ATR algorithms, the average percentage data increase with respect to the I/CATR algorithm is 27% and 10%. Similarly, on scaling the network to 800x800 meters with 3032 nodes the average percentage data increase with respect to the I/CATR algorithm is 41% and 24%. This clearly shows that the proposed algorithms are scalable with network size. Next, the impact of in-network processing is evaluated by enabling and disabling the spatial suppression algorithms. For an 800x800 meters network running the I/CATR algorithm and with a suppression threshold set to 1 unit, the amount of data reduction percentage is 35% compared to disabling suppression. Moreover, the threshold value has an impact on the amount of data that is being suppressed. For a 700x700 meters network running the I/ATR algorithm by varying the suppression threshold from 1 to 20 units the data reduction percentage is 38%. Apart from these, the impact of sink location and contour dependence on the proposed algorithms is also evaluated. The results clearly show that the proposed solutions are efficient and are not dependant on these factors. The reduction in data reflects on the node's battery power which in turn prolongs the network lifetime.

7.2 Discussion

Contours are natural phenomena which are generally continuous and smooth. Moreover, contours are not uniformly spread throughout the network. Some contours are dynamic and change frequently compared to the others. All these natural properties of the contours are helpful in making efficient design decisions which can be seen in all the proposed

algorithms. The proposed algorithms are mainly designed for one-shot queries which suit dynamic contours. However, these algorithms can be extended to periodic queries which require constant monitoring of a phenomenon which changes less frequently. An a priori knowledge of the contour helps in improving the efficiency of the pattern-based contour detection algorithm in detecting the contours for repeated queries. The continuity and smoothness of the contour is exploited in the cluster-based query propagation algorithm to propagate the query in an efficient manner independent of the network size. It is common for the adjacent nodes to have their readings spatially correlated. These spatial correlations are taken into account while designing the cluster-based spatial suppression algorithm to efficiently remove redundant spatially correlated contour readings. Similarly, for contours that do not change often and are periodically monitored, readings might show temporal correlation. This observation is the basis for the temporal suppression algorithm. The proposed I/CATR algorithm is the most efficient algorithm compared to the other proposed algorithms because it makes use of the contour continuity and smoothness to route the data response along the contour or the aggregation tree opportunistically depending on the local efficiency. Similarly, I/CSPR algorithm combines the response with the request and routes along the contour before routing the response to the sink, but can be in applications with constant payload size.

7.3 Thesis Contributions

In summary, the main contributions of this thesis are:

- A pattern-based contour detection algorithm which increases in energy-efficiency by detecting the required contour in the WSN without flooding the network. This algorithm helps in saving the node's battery power by cutting down unnecessary message receptions if the node is away from the contour. In this thesis, two classes of pattern-based contour detection algorithms, one for small scale networks and the other for large scale networks have been proposed. Single-pattern-based contour detection uses a single pattern to detect the contour in the

network. This approach might not be successful in finding contours in large scale networks, so a multi-pattern contour detection scheme has been proposed. A simple single ray-based pattern is used to detect the contour in all the experiments as a proof-of-concept.

- Cluster-based spatial suppression algorithms has been proposed to help save resources by performing suppression of spatially correlated contour readings using spatial resolution set by the user. Spatial suppression is an in-network processing technique used to reduce unnecessary message transmissions and remove redundant contour information within the cluster before forwarding the overall aggregated response to the sink. Compression techniques are also proposed to encode minimal data in the response payload to further reduce the amount of data transmitted to the sink. Data reduction using the proposed in-network processing algorithms helps in saving the node's battery power by avoiding the node from transmitting redundant data. Interpolation techniques are proposed for accurately reconstructing the contour from the received response payload. All of these in-network processing schemes are implemented in a proof-of-concept prototype and their performance is analyzed in the simulation experiments.

- Contour data routing algorithms have been proposed which fall into two categories: shortest path routing and aggregation tree-based routing. The I/SPR algorithm is a shortest path routing algorithm where the CH aggregates the member responses and routes the data to the sink. The I/ATR and I/CATR algorithms are tree-based algorithms which route the aggregated responses along the aggregation tree to the sink and perform consolidation of response headers at the intermediate nodes. Moreover, I/CATR algorithm also routes along the contour opportunistically avoiding neighbouring clusters from forwarding contour data along different paths up the aggregation tree. Data routing using I/ATR and I/CATR algorithms help in transmitting the contour data to the destination in an energy-efficient manner. All these routing techniques are implemented in a

proof-of-concept prototypes and their performance is evaluated in the experiments with the baseline F/SPR and F/ATR algorithms.

7.4 Future Work

Some possible future research directions include:

- Contour-based WSN applications require monitoring of the external phenomenon once, periodically or continuously. Algorithms proposed in this thesis are designed for one-shot queries. Extending some of these algorithms for periodic queries needs to be studied further.

- A simple single ray-based query pattern is used to detect the contours in all the experiments. This might not be appropriate for all scenarios. Sophisticated schemes such as multi-pattern and raster scan based contour detection algorithms may need to be used. In the future, these algorithms should be implemented and their performance in terms of contour detection, reliability and efficiency evaluated.

- In this thesis, spatial suppression is performed on a uniformly distributed hierarchically clustered static topology. The impact of spatial suppression on a randomly distributed clustered topology remains to be studied. The effect of temporal suppression on energy savings while using periodic contour queries should be considered as well.

- This thesis provides efficient solutions to contour-based sensor applications. However, the query propagation, in-network processing and response routing algorithms proposed can be used in other WSN applications pertaining to object tracking in the network or information retrieval from a particular geographical

area in the network. A complete protocol for these types of applications could be obtained by extending the current algorithms.

REFERENCES

[1] H. Karl and A. Willig. Protocols and architectures for wireless sensor networks. *John Wiley & Sons*, May 2005.

[2] X. Meng, T. Nandagopal, L. Li and S. Lu. Contour maps: monitoring and diagnosis in sensor networks. *International Journal of Computer and Telecommunications Networking,* Vol. 50, No. 15, October 2006, pp. 2820-2838.

[3] X. Zhu, R. Sarkar, J. Gao and J.S.B. Mitchell. Light-weight contour tracking in wireless sensor networks. *IEEE Conference on Computer Communications*, Phoenix, AZ, April 2008, pp. 1175–1183.

[4] W. Xue, Q. Luo, L. Chen, and Y. Liu. Contour map matching for event detection in sensor networks. In *Proceedings of ACM SIGMOD International Conference*, Chicago, IL, June 2006, pp. 145-156.

[5] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci. Wireless sensor networks: a survey. *IEEE. Communications Magazine*, Vol. 40 No. 8, 2002, pp. 102-104.

[6] J. M. Hellerstein, W. Hong, S. Madden, and K. Stanek. Beyond average: toward sophisticated sensing with queries. In *Proceedings of Information Processing in Sensor Networks*, Palo Alto, CA, April 2003, pp. 63–79.

[7] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, Vol. 36, January 1957, pp. 1389-1401.

[8] P. M. Wightman and M. A. Labrador. A3: a topology control algorithm for wireless sensor networks. In *Proceedings of IEEE Global Telecommunications Conference*, New Orleans, LA, November 2008, pp. 346-351.

[9] Y. Zeng, X. Jia and H. Yanxiang. Energy efficient distributed connected dominating sets construction in wireless sensor networks. In *Proceedings of International Wireless Communications and Mobile Computing Conference,* Vancouver, Canada, July 2006, pp. 797–802.

[10] J. Wu, F. Dai. An extended localized algorithm for connected dominating set formation in ad hoc wireless networks. *IEEE Transactions on Parallel and Distributed Systems,* San Francisco, CA, September 2004, pp. 908-920.

[11] J. Wu, H. Li. On calculating connected dominating set for efficient routing in ad hoc wireless networks. In *Proceedings of ACM International Workshop on*

Discrete Algorithms and Methods for Mobile Computing and Communications, Seattle, WA, August 1999, pp. 7–14.

[12] M. Gerla and J. Tsai. Multicluster, mobile, multimedia radio network. *ACM Baltzer Journal of Wireless Networks*, Vol. 1, No. 3, October 1995, pp. 255-265.

[13] D. J. Baker and A. Ephremides. A distributed algorithm for organizing mobile radio telecommunication networks. In *Proceedings of International Conference on Distributed Computer Systems*, Paris, France, April 1981, pp. 476-483.

[14] W. Heinzelman, A. Chandrakasan and H. Balakrishnan. Energy-efficient communication protocol for wireless micro sensor networks. In *Proceedings of International Conference on System Sciences*, Maui, Hawaii, January 2000, pp. 3005-3014.

[15] M. Chatterjee, S. K. Das, and D. Turgut. WCA: a weighted clustering algorithm for mobile ad hoc networks. *IEEE Journal of Cluster Computing*, Vol. 5, No. 2, April 2002, pp. 193-204.

[16] N. Li, J. C. Hou and L. Sha. Design and analysis of an MST-based topology control algorithm. In *Proceedings of IEEE conference on Computer Communications*, San Francisco, CA, March 2003, pp. 1702–1712.

[17] A. C. Yao. On constructing minimum spanning trees in K-dimensional spaces and related problems. *Society for Industrial and Applied Mathematics Journal on Computing*, Vol.11, No. 4, November 1982, pp. 721–736.

[18] D. M. Blough, M. Leoncini, G. Resta and P. Anti. The *K*-neigh protocol for symmetric topology control in ad hoc networks. In *Proceedings of Mobile Ad Hoc Networking and Computing*, Annapolis, MD, June 2003, pp. 141-152.

[19] Y. Yao and J. Gehrke. Query processing for sensor networks. In *Proceedings of Conference on Innovative Data Systems Research*, Asilomar, CA, January 2003, pp. 233-244.

[20] S. R. Madden, M. J. Franklin, J. M. Hellerstein and W. Hong. TinyDB: an acquisitional query processing system for sensor networks. *ACM Transcations on Database Systems*, Vol. 30, No. 1, March 2005, pp. 122-173.

[21] T. He, B. M. Blum, J. A. Stankovic, and T. F. Abdelzaher. AIDA: Application Independent Data Aggregation in wireless sensor networks. *ACM Transactions on Embedded Computing System*, Vol. 3, No. 2, May 2004, pp. 426–457.

[22] S. S. Pradhan, J. Kusuma, and K. Ramchandran. Distributed compression in a dense microsensor network. *IEEE Signal Processing Magazine*, Vol. 19, No. 2, March 2002, pp. 51–60.

[23] J. N. Al-Karaki and A. E. Kamal Routing techniques in wireless sensor networks: a survey. *IEEE Wireless Communications,* Vol. 11, No. 6, December 2004, pp. 6-28.

[24] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing for mobile computers. In *Proceedings of ACM SIGCOMM Conference on Communications Architectures, Protocols and Applications*, London, United Kingdom, April 1994, pp. 234–244.

[25] D. Johnson, Y. Hu and D. Maltz. The dynamic source routing for mobile ad hoc networks for IPv4. *Internet RFC 4728*, February 2007.

[26] H. Takagi and L. Kleinrock. Optimal transmission ranges for randomly distributed packet radio terminals. *IEEE Transactions on Communications*, Vol. 32, No. 3, March 1984, pp. 246-257.

[27] J. H. Chang and L. Tassiulas. Maximum lifetime routing in wireless sensor networks. *IEEE/ACM Transactions on Networking,* Vol. 12, No. 4, August 2004, pp. 609-619.

[28] C. Intanagonwiwat, R. Govindan and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of International Conference on Mobile Computing and Networks*, Boston, MA, August 2000, pp. 56-67.

[29] J. Kulik, W. R. Heinzelman, and H. Balakrishnan. Negotiation-based protocols for disseminating information in wireless sensor networks. In *Transactions on Wireless Networks,* Vol. 8, No. 2-3, March-May 2002, pp. 169-185.

[30] D. Braginsky and D. Estrin. Rumor routing algorithm for sensor networks. In *Proceedings of ACM International Workshop on Wireless Sensor Networks and Applications*, Atlanta, GA, September 2002, pp. 22-31.

[31] N. Sadagopan, B. Krishnamachari, and A. Helmy. Active query forwarding in sensor networks. *Journal of Ad Hoc Networks,* Vol. 3, No. 1, January 2005, pp. 91-113.

[32] M. Chu, H. Haussecker, and F. Zhao. Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks. *International Journal of High Performance Computing Applications*, Vol. 16, No. 3, August 2002, pp. 293-313.

[33] J. Liu, F. Zhao, and D. Petrovic. Information-directed routing in ad hoc sensor networks. *IEEE Journal on Selected Areas in Communications*, Vol. 23, No. 4, April 2005, pp. 851–861.

[34] J. Faruque and A. Helmy. Rugged: routing on fingerprint gradients in sensor networks. In *Proceedings of IEEE/ACS International Conference on Pervasive Services,* Beirut, Lebanon, July 2004, pp. 179-188.

[35] R. Sarkar, X. Zhu, J. Gao, L. J. Guibas and J. S. B. Mitchell. Iso-contour queries and gradient routing with guaranteed delivery in sensor networks. In *Proceedings of IEEE Conference on Computer Communications*, Phoenix, AZ, April 2008, pp. 1633-1641.

[36] Y. J. Zhao, R. Govindan and D. Estrin. Residual energy scan for monitoring sensor networks. In *Proceedings of IEEE Wireless Communications and Networking Conference*, Orlando, FL, March 2002, pp. 356-362.

[37] C. Buragohain, S. Gandhi, J. Hershberger, and S. Suri. Contour approximation in sensor networks. In *Proceedings of IEEE Distributed Computing in Sensor Systems*, San Francisco, CA, June 2006, pp. 356-371.

[38] I. Solis and K. Obraczka. Efficient continuous mapping in sensor networks using isolines. In *Proceedings of International Conference on Mobile and Ubiquitous Systems: Networking and Services*, San Diego, CA, July 2005, pp. 325-332.

[39] Y. Liu and M. Li. Iso-Map: energy-efficient contour mapping in wireless sensor networks. In *Proceedings of International Conference on Distributed Computing Systems,* Toronto, Canada, June 2007, pp. 311-318.

[40] M. Li, Y. Liu, "Iso-Map: Energy-Efficient Contour Mapping in Wireless Sensor Networks. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 22, No. 5, May 2010, pp. 699-710.

[41] M. Singh, A. Bakshi, and V. K. Prasanna. Constructing topographic maps in networked sensor systems. In *Proceedings of Workshop on Algorithms for Wireless and Mobile Networks*, Boston, MA, August 2004.

[42] S.Yoon, and C.Shahabi. Exploiting spatial correlation towards energy efficient clustered aggregation technique. In *Proceedings of IEEE International Conference on Communications*, Seoul, Korea, May 2005, pp. 3307-3313.

[43] S. Pattem, B. Krishnamachari, and R. Govindan. The impact of spatial correlation on routing with compression in wireless sensor networks. In *Proceedings of International Symposium on Information Processing in Sensor Networks* , Berkeley, CA, April 2004, pp. 28-35.

[44] Z. Cheng and W. Michael. Continuous contour mapping in sensor networks. *IEEE Consumer Communications and Networking Conference*, Las Vegas, NV, January 2008, pp. 152-156.

[45] The Network Simulator –ns-2. http://www.isi.edu/nsnam/ns

A. APPENDIX

A.1 Timer Types

A.1.1 CM/GN Query Response Receive Wait Timer

The CM/GN query response receive wait timer (cstCmGnQryRsRcvWtTmr) is started at the CMs/GNs on receiving the query request from their CH. This timer is used by the members to transmit their query responses to the CH on expiry. The timer value can be set to any arbitrary constant, but should be less than the sink query response receive wait timer. All nodes in the network should use the same arbitrary constant timer value which can be pre-configured or transmitted in the query. In our case, the timer value is pre-configured in all the simulations. If suppression is enabled, the timer value set at a particular node takes the hop distance to the CH into consideration. The greater the hop distance to the CH, the smaller the timer value and vice versa. However, the CMs that detect a contour from the broadcasted CH sensed reading have a greater timer value compared to the nodes that haven't detected a contour even if they are at the same hop level. The arbitrary constant timer value should be chosen in such a way that the difference in timer values between two consecutive hops is large enough to do the necessary processing at the nodes. If the suppression is disabled, then all the members set the timer to the same value.

A.1.2 Sink Query Response Receive Wait Timer

The sink query response receive wait timer (cstSnkQryRsRcvWtTmr) is started at the CH on broadcasting the query request to its members. The timer operation varies depending upon the algorithms. In the F/SPR, F/ATR and I/SPR algorithms, the CH transmits the aggregated member query responses as sink query response to the sink on timer expiry.

On the other hand, in the I/ATR and I/CATR algorithms CHs start the CH aggregate data synchronization timer on timer expiry. The timer value can be set to any arbitrary constant, but should be greater than CM/GN query response receive timer. All CHs in the network should use the same arbitrary constant timer value which can be pre-configured or transmitted in the query. In our case, the timer value is pre-configured in all the simulations.

A.1.3 CH Aggregate Data Synchronization Timer

The CH aggregate data synchronization timer (cstChAggDataSyncTmr) is used only by the I/ATR and I/CATR algorithms. It is started on sink query response receive wait timer expiry at the CHs that have detected the presence of a contour in their cluster from the member responses. This timer is used for synchronizing the clusters that lay along the contour before the CHs aggregate their member responses and forward the sink query response to the sink using the aggregation tree. This timer can be any arbitrary constant value which is dependant on the transmission time between the clusters and number of clusters along the contour. The arbitrary constant timer value can be pre-configured or transmitted in the query. In our case, the timer value is pre-configured in all the simulations. The transmission delay between clusters is calculated and pre-configured. The number of clusters along the contour that need to be synchronized is indicated by the synchronization counter and current synchronization counter parameters explained in section A.2.1 and section A.2.2. These parameters give user the flexibility to control the length of the contour that needs to be tracked.

A.1.4 Aggregate Data Wait Timer

The aggregate data wait timer (cstAggDataWtTmr) is used by the I/ATR and I/CATR algorithms. This timer is started on CH aggregate data synchronization timer expiry or when a node in the aggregation tree receives the sink query response from its children.

The timer value can be set to any arbitrary constant. All nodes in the network should use the same arbitrary constant timer value which can be pre-configured or transmitted in the query. The arbitrary constant timer value should be chosen in such a way that the difference in timer values between two consecutive hops is significant enough to do the necessary processing at the nodes. If the timer is started by CH aggregate data synchronization timer expiry then the timer value is dependant on the CH's hop distance to the sink and also the hop distance from the CH to the farthest boundary node in the WSN. The hop distance of the furthest node in the network to the sink is indicated by the maximum hop distance parameter explained in section A.2.1 and section A.2.2. If the timer is started at a node in the aggregation tree on receiving the sink query response then the timer value is just dependant on the node's hop distance to the sink. On timer expiry, the responses are aggregated in the sink query response and forwarded to the sink using the aggregation tree.

A.2 Packet Types

A.2.1 Sink Query Request

The sink query request is used by the sink to propagate the interest into the network. It unicasts the request to the neighbouring CH. Depending on the algorithms used the sink query request is processed in different ways. In the F/SPR and F/ATR algorithms, the sink query request is used as a trigger to initiate flooding of the query request in the network. It is propagated between the sink and the neighbouring CH, which on receiving the sink query request, changes the packet type to CH query request and initiates the flooding process. On the other hand, in the I/SPR, I/ATR and I/CATR algorithms the sink query request is used by the pattern-based algorithms to propagate the query request along a particular pattern in the network. Any CH that lies on the pattern receives the sink query request and on receiving the request broadcasts it within the cluster indicating specific GNs to forward the request to the neighbouring clusters along the pattern. The

GNs on receiving the sink query request check if they are destined to forward the packet to the neighbouring clusters which lie on the specified pattern. Figure A.1 indicates the packet format of sink query request.

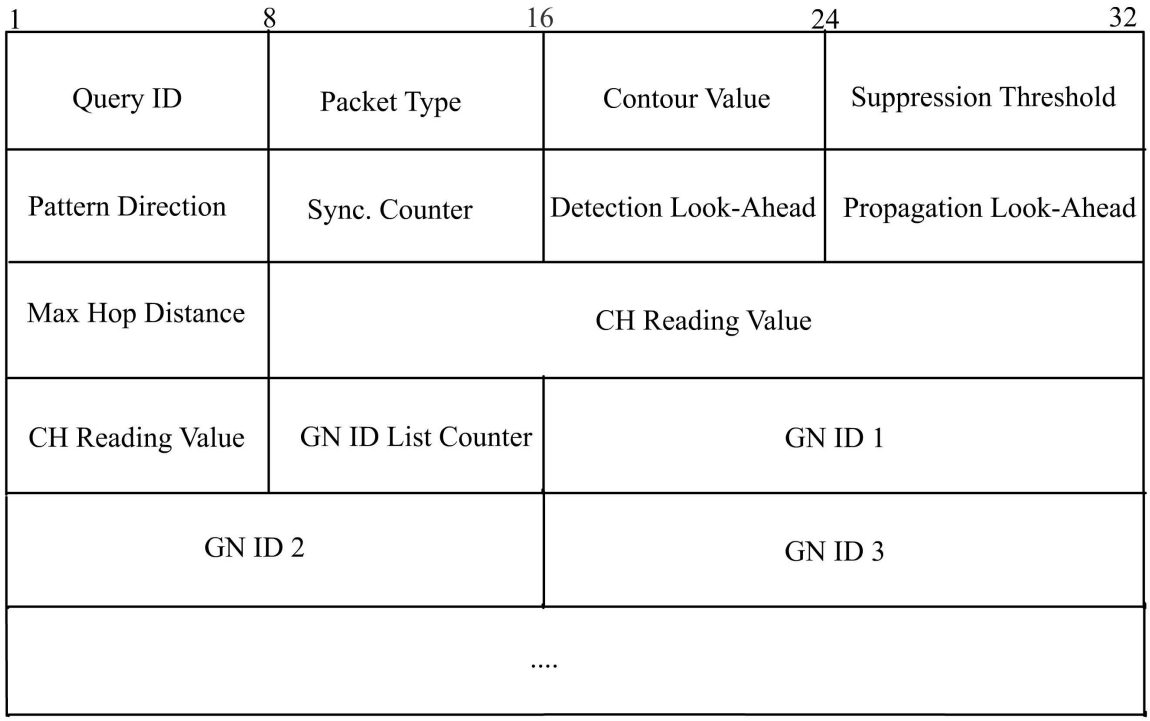

Figure A.1: Sink Query Request

The query ID is unique and identifies a particular query. Packet type indicates the type of packet that is being used. Contour value indicates the value that the sink is interested in from the network. Suppression threshold is an optional parameter used to indicate the spatial suppression resolution and is encoded in the packet only when suppression is enabled. Pattern direction specifies the pattern in which the query has to be propagated in order to detect the contour and is only encoded when the I/SPR, I/ATR and I/CATR algorithms are used. In the experiments, single ray-based pattern is used. The synchronization counter is encoded only when the I/ATR and I/CATR algorithms are used and is used to synchronize the data among the clusters before they propagate the aggregated sink query responses to the sink using the aggregation tree. Detection look-

ahead is used only by the I/SPR, I/ATR and I/CATR algorithms to increase the detection band size along the pattern. Similarly, propagation look-ahead is used by these algorithms to increase the propagation band size along the contour as explained in section A.2.2. In all the experiments, the detection look-ahead parameter is not encoded in the packet because a constant look-ahead of one is considered. Maximum hop distance is used only by the I/ATR and I/CATR algorithms to set the constant aggregation data wait timer at the CHs which have detected the contour. It gives the maximum node hop distance to the sink in the WSN. CH reading value is the reading sensed by the CH and is used by the members within the cluster to detect a contour between themselves and the CH. The GN ID list is the count of GN IDs that are encoded in the packet. The encoded GN IDs are required to forward the sink query request to the neighbouring clusters along the ray.

A.2.2 CH Query Request/ Forward CH Query Request

The CH query request is used by the CH to detect the presence of a contour passing through the cluster. The request is broadcast by the CH within the cluster. Irrespective of whether a contour is detected within a cluster or not in the F/SPR and F/ATR algorithms propagate the CH query request to the neighbouring CHs. On the other hand, the I/SPR, I/ATR and I/CATR algorithms broadcast the CH query request within the cluster. Only after receiving the query responses from the members indicating the presence of a contour does the CH broadcast the query to the neighbouring clusters using forward CH query request. GNs on receiving the forward CH query request change the request type to CH query request and forward it to the neighbouring CH along the contour. Figure A.2 indicates the packet format of CH query request/forward CH query request.

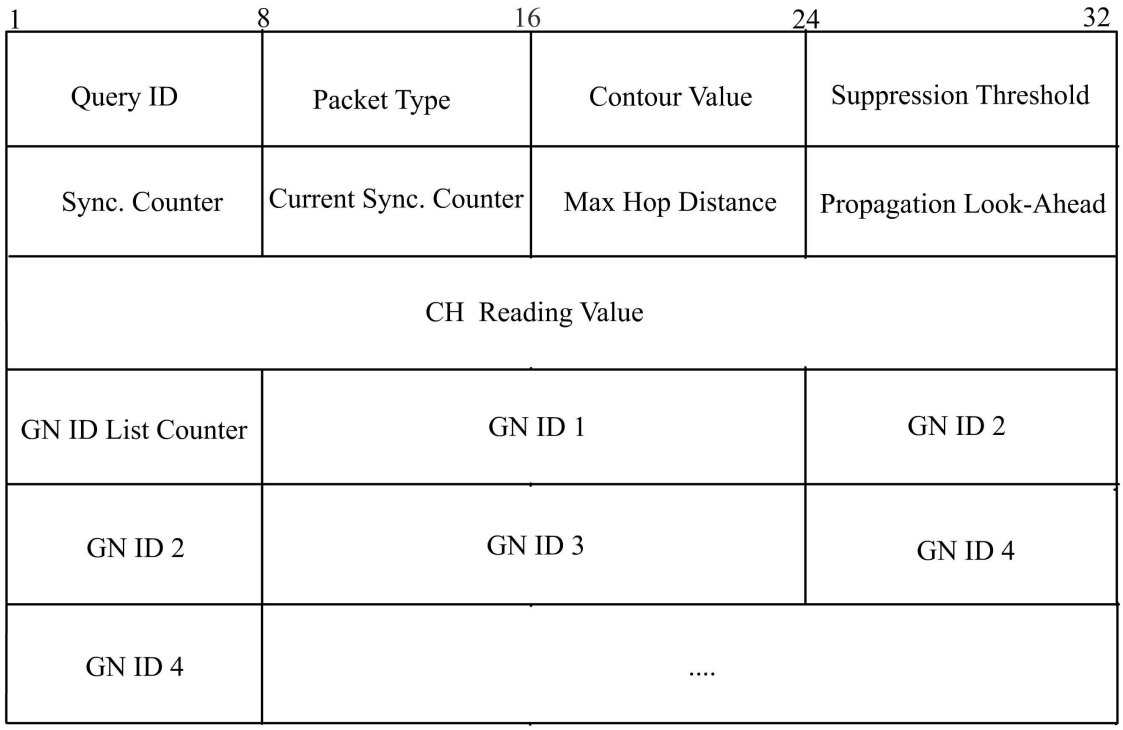

Figure A.2: CH query request/ Forward CH query request packet format

In the F/SPR and F/ATR algorithms the synchronization counter, current synchronization counter, maximum hop distance and propagation look-ahead are not encoded in the CH query request. While using the I/SPR, I/ATR and I/CATR algorithms, excluding the GN ID list and the GN IDs all the other parameters included in the F/SPR and F/ATR algorithms are encoded in the CH query request (broadcasted within the cluster). In the I/ATR and I/CATR algorithms, all the parameters are encoded except the CH reading value in the forward CH query request. Similarly in the I/SPR algorithm, CH reading value is excluded and propagation look-ahead parameter is included in the forward CH query request in addition to all the parameters encoded by F/SPR and F/ATR algorithms. Propagation look-ahead is used by I/SPR, I/ATR and I/CATR algorithms to increase the propagation band size while propagating request along the contour. The current synchronization counter is the running synchronization counter value used by the CHs along the contour for synchronization of their aggregated data. GNs on receiving the forwarding CH query request remove the GN ID list and the GN IDs from the packet and

change the packet type to CH query request before forwarding to the neighbouring CHs along the contour.

A.2.3 CM/GN Reading Request

The CM/GN reading request is used by the members within the cluster to exchange their sensed readings with the neighbours. CMs/GNs broadcast the reading request to their neighbours. Figure A.3 indicates the packet format of CM/GN reading request.

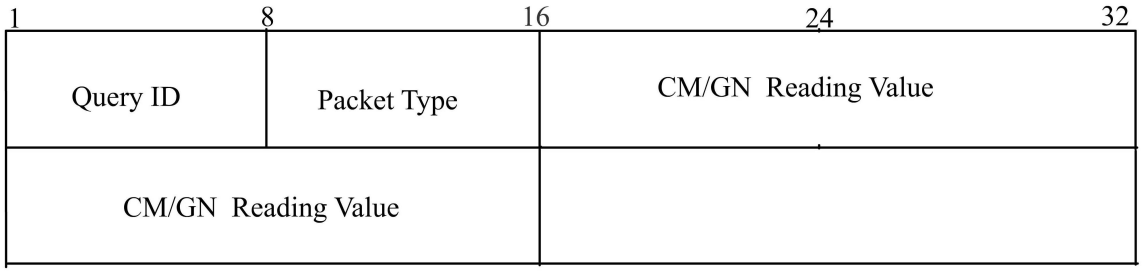

Figure A.3: CM/GN reading request packet format

A.2.4 CM/GN Query Response

The CM/GN query response is used by the members within the cluster to forward their responses to the CH. CMs/GNs broadcast their responses to the neighbours. Figure A.4 indicates the packet format of CM/GN query response.

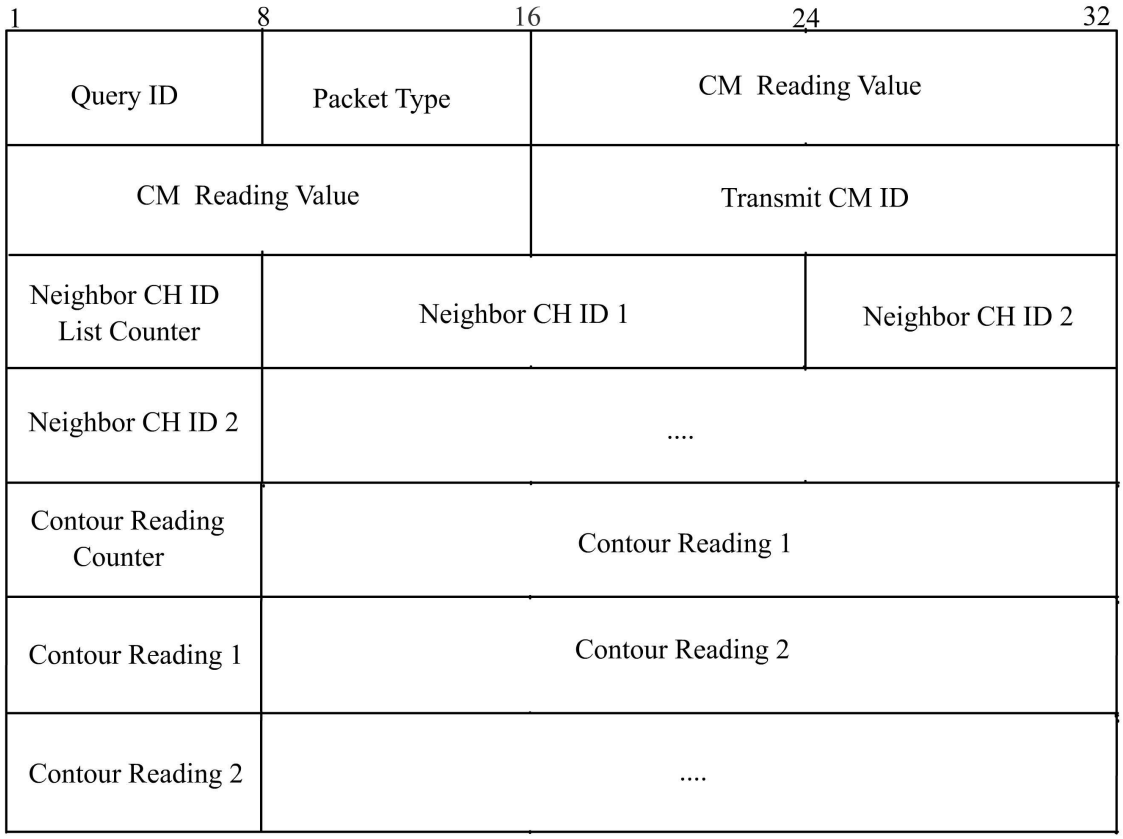

Figure A.4: CM/GN query response packet format

The CM reading value is only used by the CMs to transmit their sensed reading value. Transmit CM ID is used only by the GNs to encode the ID of the CM that needs to further forward the GN query response to the CH. Neighbour CH ID list is the count of neighbouring CH IDs that are encoded in the packet. Neighbour CH IDs are encoded only when the I/CATR algorithm is used for the CH to make decisions whether to route to along the contour or the aggregation tree to the sink. These IDs are encoded by the nodes only when they detect a contour with the nodes in the neighbouring clusters. The contour reading counter gives the count of contour readings that are encoded in the packet. Contour readings contain the necessary contour information for the sink to reconstruct the contour at the sink.

A.2.5 Sink Query Response

The sink query response is used by the nodes to forward their response to the sink. Nodes forward their response using unicast. Figure A.5 indicates the packet format of sink query response.

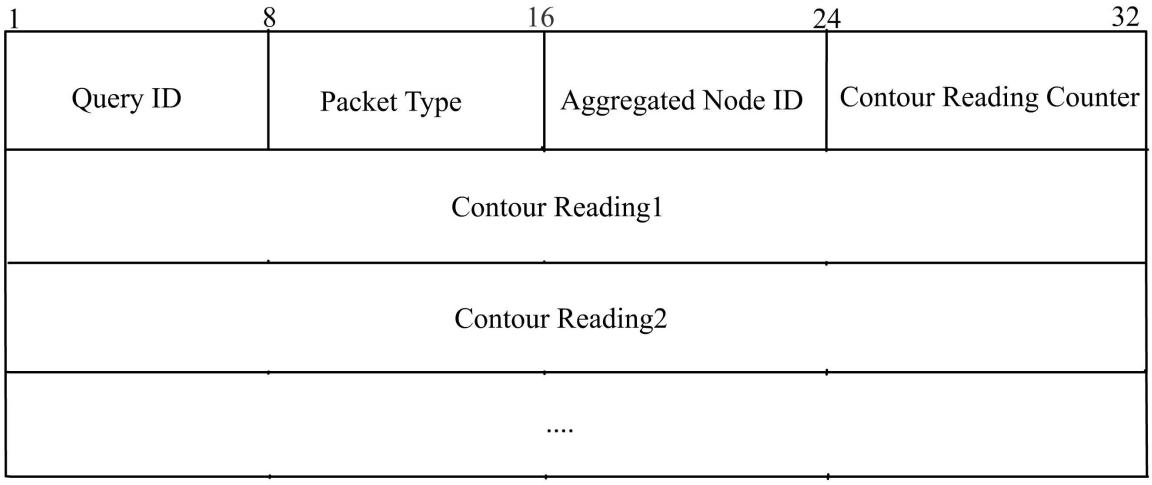

Figure A.5: Sink query response packet format

A.2.6 Change Parent CH ID Request

The change parent CH ID request is used by the I/CATR algorithm for changing the CH ID of a parent. This packet is forwarded by using unicast. Figure A.6 indicates the packet format of change parent CH ID request.

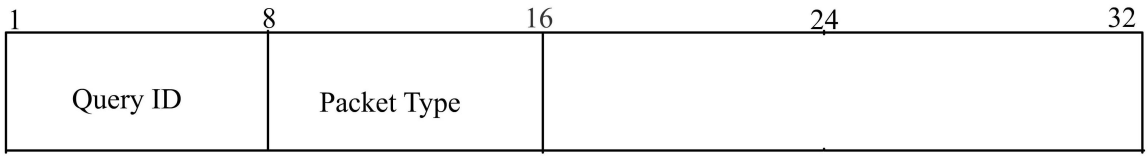

Figure A.6: Change parent CH ID request packet format