Multi-Sink Mobile Wireless Sensor Networks:

Dissemination Protocols, Design and Evaluation

Ayşegül Tüysüz Erman

Multi-Sink Mobile Wireless Sensor Networks: Dissemination Protocols, Design and Evaluation

Ayşegül Tüysüz Erman

This research was conducted within the EU project AWARE (IST-2006-33579) and Point.One project RECONSURVE-NL.

Pervasive Systems Research Group
Faculty of Electrical Engineering, Mathematics and Computer Science
University of Twente, The Netherlands.

CTIT PhD Thesis Series Number 11-206
ISSN 1381-3617
Center for Telematics and Information Technology
P.O. Box 217, 7500 AE Enschede, The Netherlands.

Keywords: Wireless sensor networks, mobile multi-sink, mobile sensors, data dissemination, query dissemination.

Cover Photo: Folashade Adeyosoye; Bees on Honeycomb
Cover Design: Kamil & Ayşegül Erman

Printed by Wöhrmann Print Service, Zutphen, The Netherlands.

ISBN 978-90-365-3251-8
http:/dx.doi.org/10.3990/1.9789036532518

MULTI-SINK MOBILE WIRELESS SENSOR NETWORKS: DISSEMINATION PROTOCOLS, DESIGN AND EVALUATION

DISSERTATION

to obtain
the degree of doctor at the University of Twente,
on the authority of the rector magnificus,
prof.dr. H. Brinksma,
on account of the decision of the graduation committee,
to be publicly defended
on Thursday the 1st of September 2011 at 16.45

by

Ayşegül Tüysüz Erman

born on 17 June 1980,
in Bahçelievler, Istanbul, Turkey

This dissertation is approved by:

Prof. Dr. Paul J.M. Havinga (promotor)

Dr. Arta Dilo (assistant promotor)

Abstract

In pervasive systems, as they are getting smaller and smaller, computers can be found just about everywhere, but their presence is not noticed because the technologies are often embedded within items. One of the smallest and well known embedded computers is a wireless sensor node, which is a passive sensing device capable of communicating wirelessly with other devices. Early attempts to monitor the physical environment are primarily composed of these passive sensing devices which have been succeeded in many applications by the development of Stationary Wireless Sensor Networks. A wireless sensor network is typically composed of many tiny computers, often no bigger than a coin or a credit card, that feature a low frequency processor, some flash memory for storage, a radio for short-range wireless communication, on-chip sensors and an energy source such as AA batteries. Applications of stationary wireless sensor networks have emerged in many domains ranging from environmental monitoring to structural monitoring as well as industry manufacturing.

In all these applications, the primary task of a wireless sensor network is to collect useful information by monitoring phenomena in the surrounding environment. Typically, in a wireless sensor network, sensor nodes generate data about a phenomenon and relay streams of data to a more resource rich device, namely a data sink, for analysis and processing. Early sensor networks have been modeled as having a single, predefined, stationary sink. However, as the size of the sensor network grows with the wide availability of economically viable embedded sensor nodes, the communication between the sensors and the single stationary sink can lead to high energy consumption, and consequently reduce the lifetime of the network. In recent years there has been renewal of interest in using multiple sinks for wireless sensor networks to achieve power saving. Although multi-sink partitioning of the sensed area enhances some performance metrics, such as network lifetime, of single sink sensor network, the development of multiple stationary sinks in an area of interest still creates an uneven energy depletion phenomenon around the sinks, since sensors near a data sink deplete their battery power faster than those far apart, due to their heavy overhead of relaying messages.

The improvements of stationary wireless sensor networks in conjunction with the advances developed by the distributed robotics and low power embedded systems communities have led to a new class of Mobile Wireless Sensor Networks that can be utilized for a wide range of scenarios such as land, sea and air exploration and monitoring, habitat monitoring, vehicular applications, and emergency response, which require reliable and timely collection of data. Current studies have tried to utilize the advantages of mobile sensors to overcome the problems of stationary sensor networks. Mobile Wireless Sensor Networks have a similar architecture to their stationary counterparts, thus are governed by the same energy and processing limitations, but require the development of a new generation of algorithms targeting at constantly changing network topologies due to sink and/or sensor mobility.

This thesis focuses on the efficient data extraction and dissemination in wireless sensor networks by making use of the multiple sinks and by handling mobility of sensors and sinks. We start with analyzing the characteristics of multi-sink wireless sensor networks. We propose a set of algorithms that enable multi-sink wireless sensor network to self-organize efficiently in the presence of mobility and adapt to dynamics in order to increase the functionality of the network. Our contributions include an algorithm for load balancing in multi-sink sensor networks, a protocol for query dissemination towards an area of interest combined with a set of algorithms that are used to handle mobility efficiently in a tree-based routing, and a data dissemination protocol that tackles sink mobility in a wireless sensor network. In short, the main contributions of the thesis are listed as follows:

✓ ***Benefits and challenges of using multiple sinks in static wireless sensor networks:*** We review the state of the art multi-sink partitioning methods in wireless sensor networks. We present a multi-sink partitioning mechanism to achieve load balancing between sinks.

✓ ***Design and evaluation of a query dissemination protocol for multi-sink wireless sensor networks:*** To enable sinks to efficiently route queries which are only valid in particular regions of the deployment, we propose a set of algorithms which combine coverage area reporting and geographical routing of queries that are injected by sinks.

✓ ***Handling mobility of sensors in a tree-based dissemination protocol:*** To provide an up-to-date coverage area description to sinks, we focus on handling sensor node mobility in the network. We discuss what is the best method to handle mobility in tree-based routing of queries: (i) periodic global updates from every sink or (ii) local updates only from mobile sensors. We propose a method to achieve local updates which are needed to handle sensor mobility in a tree-based network.

✓ ***Design and analysis of a data dissemination protocol for mobile multi-sink sensor networks:*** To achieve reliable data dissemination of events as well as the efficiency in handling the mobility of both multiple sinks and event sources, we propose a virtual infrastructure and a data dissemination protocol, namely HexDD (Hexagonal cell-based Data Dissemination) exploiting this infrastructure. We analytically compare the communication cost and hot region traffic cost of the proposed data dissemination with other approaches.

✓ ***Evaluation of the data dissemination protocol for different wireless sensor network applications:*** We focus on the performance evaluation of the data dissemination protocol, HexDD, in two different classes of mobile wireless sensor networks: (i) mostly static, which contains scenarios in which most of the sensors are static and some sensors are attached to people or vehicles such as firefighters or unmanned aerial vehicles moving at low or medium velocities in an Emergency Response Application, (ii) highly mobile, which contains scenarios in which many sensors are attached to devices that move at high velocities such as cars in a Vehicular Sensor Network Application.

Samenvatting

Nu pervasive systemen steeds kleiner worden, kun je overal computers tegenkomen. Maar omdat de technologie in de praktijk vaak ingebed is in andere producten, worden ze meestal niet opgemerkt. Een van de kleinste en bekendste embedded computers is een draadloze sensornode, een passief sensorapparaat dat draadloos kan communiceren met andere apparaten. Waar vroegere systemen voor observatie van de fysieke omgeving hoofdzakelijk bestonden uit zulke passieve sensorapparaten, zijn deze in veel huidige toepassingen achterhaald door de ontwikkeling van stationaire draadloze sensornetwerken. Een typisch draadloos sensornetwerk bestaat uit vele miniatuurcomputers, vaak niet groter dan een bankpasje of een geldstuk. Zulke computertjes hebben een laagfrequente processor, een beetje flashgeheugen voor opslag, een radio voor draadloze communicatie over korte afstand, on-chip sensoren, en een energiebron, bijvoorbeeld AA-batterijen. Stationaire draadloze sensornetwerken hebben hun weg gevonden in vele toepassingsgebieden, van milieu-observatie tot structural monitoring (conditiebewaking van constructies) en toepassingen in de productie-industrie.

In al deze toepassingen bestaat de hoofdtaak van een draadloos sensornetwerk uit het verzamelen van nuttige informatie door middel van observatie van verschijnselen in de omgeving. In een typisch draadloos sensornetwerk genereren de sensornodes gegevens over een verschijnsel en sturen die gegevens in stromen door aan een krachtiger apparaat, namelijk een datasink, voor analyse en verwerking. Vroeger werden sensornetwerken gemodelleerd met een enkele voorgedefinieerde stationaire sink. Echter, nu de beschikbaarheid van ingebedde sensornodes toeneemt en tegelijk de kosten afnemen, worden sensornetwerken steeds groter, waardoor de communicatie tussen de sensoren en de enige stationaire sink tot een hoog energieverbruik kan leiden, en daarmee de levensduur van het netwerk verkort wordt. De laatste jaren is er hernieuwde belangstelling voor het gebruik van meerdere sinks voor draadloze sensornetwerken om energie te besparen. Hoewel een multi-sinkverdeling van de geobserveerde omgeving bepaalde eigenschappen, zoals de netwerklevensduur, van een single-sink-sensornetwerk verbetert, leidt de inzet van meerdere stationaire sinks in het observatiegebied nog steeds tot een onevenwichtig energieverbruik rondom de sinks. De sensoren in de buurt van een sink verbruiken hun batterijvermogen namelijk sneller dan die die zich verder weg bevinden, omdat de nabije sensoren veel meer databerichten ontvangen en doorgeven richting sink.

De verbeteringen op het gebied van stationaire draadloze sensornetwerken hebben samen met de vooruitgang bij de gedistribueerde robotica en energiezuinige embedded systemen geleid tot een nieuwe klasse van mobiele draadloze sensornetwerken, die ingezet kunnen worden in een breed scala van scenario's waar betrouwbare en snelle gegevensverzameling vereist is, zoals verkenning en observatie te land, ter zee en in de lucht, het monitoren van leefgebieden, toepassingen in voertuigen, en voor reddingswerk en hulpdiensten. Huidige studies proberen gebruik te maken van de voordelen van mobiele sensoren om de problemen van stationaire sensornetwerken te vermijden. Mobiele draadloze sensornetwerken hebben een architectuur die vergelijkbaar is met die van stationaire, en hebben dus te maken met dezelfde beperkingen ten aanzien van energieverbruik en rekenkracht. Ze vereisen echter wel de ontwikkeling van een nieuwe generatie algoritmen die zich richt op de almaar veranderende netwerktopologie ten gevolge van de mobiliteit van sinks en/of sensoren.

Dit proefschrift concentreert zich op de efficiënte extractie en verspreiding van gegevens in draadloze sensornetwerken door gebruik te maken van meerdere sinks en door de mobiliteit van sensoren en sinks aan te pakken. We beginnen met de analyse van de karakteristieke eigenschappen van multi-sink draadloze sensornetwerken. We stellen een verzameling algoritmen voor die een multi-sink draadloos sensornetwerk in staat stellen zichzelf bij mobiliteit efficiënt te organiseren en zich aan te passen aan veranderingen om zo de functionaliteit van het netwerk te verbeteren. Onze bijdragen zijn onder andere een algoritme voor multi-sink-partitionering, een protocol voor de disseminatie van een query (zoekvraag) naar een interessegebied gecombineerd met een verzameling algoritmen die gebruikt kunnen worden om mobiliteit efficiënt aan te pakken in routering met een boomstructuur, en een datadisseminatieprotocol dat sink-mobiliteit mogelijk maakt in een draadloos sensornetwerk. Samengevat zijn de belangrijkste bijdragen van dit proefschrift als volgt:

✓ ***Voordelen en uitdagingen van het gebruik van meerdere sinks in statische draadloze sensornetwerken:*** We geven een overzicht van de state-of-the-art van multi-sink partitioneringsmethoden in draadloze sensornetwerken. We presenteren een multi-sink-partitioneringsmechanisme om een evenwichtige verdeling van de belasting van de verschillende sinks te bereiken.

✓ ***Ontwerp en evaluatie van een query-disseminatieprotocol voor multi-sink draadloze sensornetwerken:*** Om sinks de mogelijkheid te bieden een efficiënte routering te maken voor query's die slechts geldig zijn in bepaalde gebieden van het netwerk, stellen we een verzameling algoritmen voor die de rapportage van het dekkingsgebied combineren met de geografische routering van query's afkomstig van sinks.

✓ ***Mobiliteit van sensoren afhandelen met een disseminatieprotocol met een boomstructuur:*** Om een actuele beschrijving van het dekkingsgebied aan de sinks te leveren concentreren we ons op de afhandeling van de mobiliteit van sensornodes binnen het netwerk. We bespreken wat de beste methode is om mobiliteit af te handelen in query-routering met een boomstructuur: (i) periodieke globale updates van iedere sink, of (ii) enkel lokale updates van mobiele sensoren. We stellen een methode voor om te komen tot lokale updates die nodig zijn voor de aanpak van sensormobiliteit in een netwerk met een boomstructuur.

✓ ***Ontwerp en analyse van een datadisseminatieprotocol voor mobiele multi-sink sensornetwerken:*** Om te komen tot niet alleen een betrouwbare datadisseminatie van events (gebeurtenissen), maar ook een efficiënte afhandeling van de mobiliteit van zowel meerdere sinks als bronnen van events, stellen we een virtuele infrastructuur voor, en een datadisseminatieprotocol, namelijk HexDD (Hexagonal cell-based Data Dissemination), dat gebruikmaakt van deze infrastructuur. We vergelijken de communicatiekosten en de “hot region” verkeerskosten van de voorgestelde datadisseminatie analytisch met andere benaderingen.

✓ ***Evaluatie van het datadisseminatieprotocol voor verschillende toepassingen van sensornetwerken:*** We richten ons op de evaluatie van de prestaties van de datadisseminatie, HexDD, in twee verschillende klassen van mobiele draadloze sensornetwerken:

(i) voornamelijk statisch; dit omvat scenario's waarin de meeste sensoren stationair zijn en enkele sensoren bevestigd zijn aan mensen of voertuigen, zoals brandweerlieden of onbemande luchtvaartuigen met lage of middelmatige snelheid in een toepassing voor hulpdiensten, en (ii) zeer mobiel; dit omvat scenario's waarin veel sensoren bevestigd zijn aan objecten die met hoge snelheid bewegen, zoals auto's in een toepassing van sensornetwerken in voertuigen (Vehicular Sensor Networks).

Acknowledgements

Looking back, I am surprised and at the same time very grateful for all I have received throughout my PhD life (except for a couple of little wrinkles around my eyes) although completing the PhD degree was probably the most challenging activity of the first 30 years of my life. It definitely involved countless cycles of exploration, inquiry, meditation, enlightenment, doubt, confusion, uncertainty, and perseverance. Over the years, I have learned to be patient and never give up. If the question of ”*Do you do a PhD in your next life?*” comes, my answer will be a definite ”*YES*”! It was a great experience, full of learning, nice memories from trips to conferences, and wonderful people in an international environment. My PhD years and this thesis have had mentorship from numerous outstanding individuals both from within the university and outside of it. With pleasure and gratitude, it is time to acknowledge all those people.

First of all, I would like to express my utmost gratitude to my promotor, *Prof. Paul Havinga*, for his prompt and useful advices during my research. Although he is one of the busiest men in the world, he never said ‘no’ to me when I knocked his door for a discussion. In our discussions, I have sometimes been exposed to his challenging critics, but always with a positive, optimistic attitude and encouragement. He always helped me to see problems from a different perspective. I am grateful to him also for funding my visit to Ohio State University which was a very valuable occasion for me. Paul, thanks for all your support, especially for giving me the opportunity to live through such a great life experience.

I feel very lucky to have *Dr. Arta Dilo* as my daily supervisor during the last two years of my PhD. She is such a friendly and patient person. I have really learned a lot from her; how to present an idea properly and to construct precise statements. We even met at the weekends to discuss my work. She provided refreshing insight, critical questions and valuable comments on my papers. I am grateful for her steady encouragement and infinite attention to details that ensured the quality of my work.

During my visit to Ohio State University, our rewarding discussions with *Dr. Eylem Ekici* were very useful to learn more about theoretical aspects of wireless networks. I am grateful to him for welcoming me to his group.

I would like to thank all of the members of my graduation committee for reading my manuscript. Special thanks to *Prof. Gerard Smit*. His extremely valuable comments and suggestions helped me a lot to express my ideas better and to improve this thesis.

I was very fortunate when I first came to Twente since *Özlem*, my friend/colleague from Turkey, started her PhD in the same group two years before me. *Özlem and Mustafa* helped me a lot with settling. Mustafa built my IKEA furnitures and Özlem always gave me very useful information about the life in Twente. We really became very close friends and shared a lot of good moments in many activities. I would like to express my deepest gratitude to them for making my life easier when I was feeling homesick at the beginning of my PhD life. Our ways will cross again soon in Istanbul. Since they are now the parents of the cutest baby in the world (*Zeynep*), I will demand for information about baby care from them at this time.

Yang was a very nice officemate, with whom I enjoyed talking about everything but work. He always gave me very nice gifts after his trips to China. He even brought me a key chain from Beijing 2008 Olympic Games. Many thanks for the very warm friendship and for accepting to be my paranymph during my defence.

I am very happy and proud of making wonderful friends in Netherlands. *Raluca and Mihai*, they form a very well-matched couple, who graduated at the same day and had two beautiful sons (*Vlad and Petru*) in two years. I will always cherish the wonderful time spent together with them, especially in our Turkish-Romanian birthday parties. Many thanks for your support and friendship. *Supriyo and Anindita* with their sweet little *Samhita*, they are always so warm and gracious. I really enjoyed listening Samhita's nice stories about her trip to Singapore while Supriyo was driving me to home after our group trip to Vlieland. I will be very happy to welcome you in Istanbul. I would like to thank *Michel* for his great patience at trying to teach me Dutch and for introducing me to amazing Latin dancing. It was a pleasure to dance with him in one of the Salsa nights.

I always think that our group is very lucky since we have *Nirvana*. She has the unique ability to communicate with any type of people and her fun-loving and caring personality makes her easy to speak to. There are very few people who I feel comfortable sharing my problems with, and *Nirvana* is definitely one of them. I thank her for showing me the right ways to overcome my disappointments and confusion during my PhD.

Everyday working life would not have been so joyful without my colleagues in the Pervasive Systems group. *Majid, Alireza, Zahra, Pouria, Kyle* (thanks a lot for the traditional Hong Kong food), *Ramon* (thanks a lot for your collaboration in the book chapter), *Cagri, Bram, Stephan, Wouter, Ardjan, Marlies, Berend Jan* (Many thanks for translating the abstract of my thesis to Dutch), *Hans* (thanks for your collaboration in the book chapter), I thank you all for creating such a nice atmosphere in our group. I would also like to acknowledge some former members of our group. *Kavitha*, many thanks for your friendship and helping me to write a proper intership letter. *Lodewijk*, thank you for your help in the AWARE project and in the first two years of my PhD. I would like to thank *Thijs and Arie* who supported me in the field experiments of AWARE project in Spain. I would also like to thank our dear secretaries, *Marlous, Nicole and Thelma*, for putting a great effort to make things easier with their excellent administrative support.

Many thanks to photographer *Folashade Adeyosoye* for giving the permission to use his great photograph, 'Bees on Honeycomb', that you see on the cover of this thesis.

Although I was 2600 km away from my home county in Twente, I always felt at home with the support of my Turkish friends. The Turkish Student Association (TUSAT) has been a wonderful organization helping students and arranging many activities for both Turkish and International people of UT. I was very fortunate to be a board member of TUSAT in 2008. It was such a pleasure to work with my fellow board members and friends - *Janet, Ayşe, Emre, and Selim* - and to organize many activities for TUSAT members. Ayşe was always very helpful about everything, not only in TUSAT, but also in my life. My deepest thanks to all members of the Turkish community, *Ayşe, Hasan, Arda, Seçkin, Erhan-Arzu, Feridun-Suzan, Didem-Semih*, who are my dearest friends from the beginning of my PhD journey. Thanks a lot for your warm support and friendship. *Bilge*, many thanks for playing squash with me when I was very stressful and trying to discharge. Special thanks to *Cem and Elif* for taking care of my flowers when I was traveling. My sweet neighbor, Elif, always listened me when I was talking about my boring thesis writing progress. Cem, thanks a lot for being there whenever I needed and for being my paranymph!

The Turkish community in Twente has extended day by day. I have spent amazing times with *Pınar-Berk*, *Neşe-Fehmi* (the super sportive couple), *Özlem-Sertan*, *Imran-Akin*, *Buket-Efe*. They were more or less like our extended family in Twente. Ladies, many thanks for inspiring me about cooking (Özlem-Imran), sewing/knitting (Özlem-Pınar), and drawing/painting (Neşe-Buket). Gentlemen, thanks a lot for always playing volleyball with me (Berk-Fehmi), and for spending time to organize amazing gatherings (Sertan-Akin). Special thanks to *Yonca* for being such a wonderful friend and well-wisher to me. All our trips, parties, board games were simply unforgettable.

My parents, *Asuman and Necati Tüysüz...* Their love, committed support, and prayers in all that I have done till now are the keys of all my achievements. I would like to thank my parents for always encouraging me to go after my dreams and to strive for more. *Canım annecim ve babacım, benim bugünlere ulaşabilmem için harcadığınız emeğin karşılığını verebilmem mümkün değil. Bana verdiğiniz sonsuz sevgi ve destek için çok teşekkür ederim.* My handsome brother, *Hasan* and my beautiful sister, *Burcu*, have also been the best of friends along this journey. Many thanks for making me laugh and for being a source of moral encouragement during times of distress. *Nilay*, thank you so much for taking care of my family when I was not there.

I am grateful to God for giving me wonderful and loving parents-in-law, *Zühra and Eyüp Erman*. They always want for nothing other than our happiness. Many thanks for the nice trips we made together to the different regions of Turkey. *Zühra annecim ve Eyüp babacım, beni her zaman destekleyip, cesaretlendirdiğiniz için çok teşekkür ederim.* My sisters-in-law, *Kezban and Fatma*, and their husbands, *Recep and Sertaç*, supported us in our hardest times and helped us to solve our critical problems. Thanks a lot for being there for us. Special thanks to my sweet nephews, *Erman and Orhan*, and the little princess, *Erin*, for bringing lots of joy and fun around.

Finally, my most heartfelt thanks...

To Kamil Erman,
the friend who keeps me centered when I feel like I am being pulled in a thousand different directions, the man who can make my world seem instantly more wonderful with just a kiss, and the companion of my heart through everything life brings...

Ayşegül Tüysüz Erman
August 2011
Enschede, the Netherlands

Contents

CHAPTER I
Introduction

It could be observed that the computing paradigm is in transition and migrating towards one where computing is pervasive, being seamlessly embedded in the fabric of everyday devices [125]. Historically, this vision was first articulated by Weiser in his description of the ubiquitous computing, more commonly referred to as the pervasive computing concept [167], in the early 1990s. In this vision, people will be surrounded by unnoticeable computers embedded within items and will use these computers unconsciously to achieve everyday tasks. Sensors and Wireless Sensor Networks (WSNs) offer distinctly attractive enabling technologies for pervasive computing environments. Wireless sensor networks with their distributed sensing capabilities have attracted a wide range of disciplines, where close interactions with the physical world are essential [29]. The early research efforts to monitor the physical environment have been focused on the development of stationary wireless sensor networks having a single data collector, namely a data sink. However, with the increasing use of WSNs in different applications, ranging from habitat monitoring to emergency response for more complex functionalities, a new type of network, Mobile Wireless Sensor Network, has emerged in today's market. Mobility poses another set of unique challenges to be addressed, which include topology management, routing, and energy management. This thesis is motivated by the communication problems in multi-sink mobile WSNs. This chapter presents the general features of sensor devices and wireless sensor networks. We also explain the characteristics and challenges of communication in WSNs and present the data reporting models which led us to the research question and approach in this thesis. Finally, we introduce our contributions and conclude with the organization of the topics studied in the thesis.

1.1 Wireless Sensor Networks

A wireless sensor network is typically composed of many tiny computers called sensor nodes, often no bigger than a coin or a credit card. The primary goal of a wireless sensor network is to collect useful information by monitoring phenomena in the surrounding environment and send the information to a data collector, namely, a sink. In WSNs, each sensor node individually senses the local environment, but collaboratively achieves complex information gathering and dissemination tasks. Therefore, the objective of wireless sensor nodes is twofold: (1) obtain a description of the physical surroundings by means of sensors, and (2) wirelessly communicate this description and assist other nodes to deliver descriptions. To carry out these two functions, a wireless sensor node is typically equipped with the following components: on-chip sensor(s), transceiver, a low frequency processor, some flash memory for storage, and power supply unit. Figure 1.1, which is redrawn from [31], shows a schematic diagram of sensor node components:

- *Sensors* are responsible for sensing (measuring) the physical environment. A node can have more than one sensor measuring different phenomena on-board. The number and types of sensors vary according to the application requirements. There is a wide variety of sensors available in the market. The most typical examples of sensors are temperature, humidity, light, pressure, vibration, sound, chemical such as CO-sensor, and body sensors such as heart rate, accelerometer. The analog signals produced by

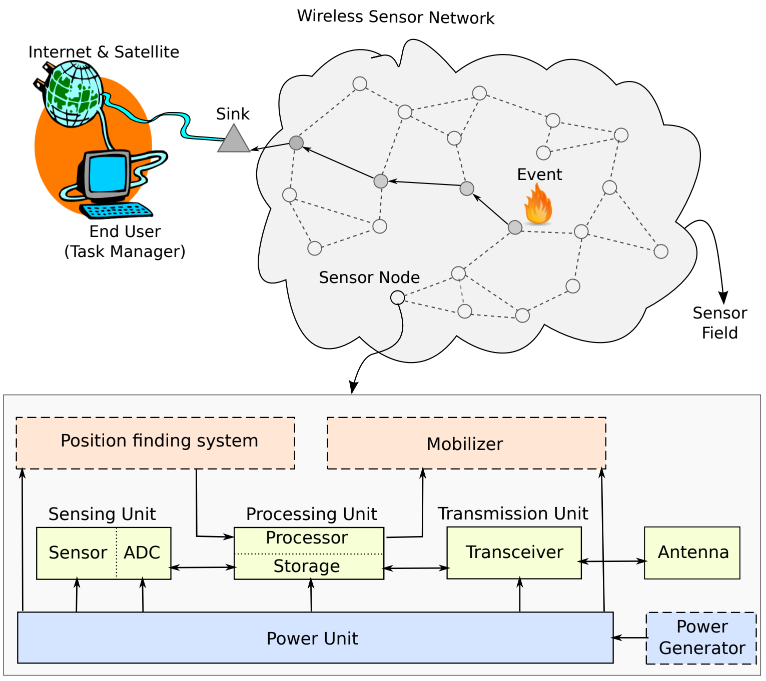


Figure 1.1: *The components of a sensor node and a wireless sensor network*

the sensor based on the observed phenomena are converted to digital signals by the ADC (analog-to-digital converter), then fed into the processing unit.

- *Wireless Transceiver* is a low-power radio for short-range wireless communication. In general, half-duplex transceivers have three operational states: *transmit*, *receive* and *standby*. Generally, transmitting consumes more power than receiving and standby lies beneath the power consumption of receiving by a factor of 1,000 or more [162]. In the literature the following bands are used for WSN applications: 433 MHz, 868 MHz, 915 MHz and 2.4 GHz, depending on which country the product is designed for. Maximum transmit powers range from 0 dBm to 13 dBm, receive sensitivity from -109 dBm to -93 dBm and bit rates from 40 kbaud to 1152 kbaud.

- *Processor* is generally required for the computation of intensive functions, like pre-processing of sensor readings, in-network processing, data aggregation, and other networking tasks. Mostly 8-bit or 16-bit processor are used. Although many of these functions can efficiently be implemented in application-specific integrated circuits (ASICs), the flexibility and ability to be reprogrammed make general processing architectures an attractive choice. Nodes usually run specialized operating systems such as TinyOS [157, 18], or AmbientRT [81] to meet the resource constraints.

- *Memory/Storage*, which has a quite limited capacities, is obviously required on the wireless sensor node to hold the program of the processor. Nowadays, many low-power micro-controllers include FLASH memory, which can be rewritten many times, yet has excellent retention properties. Often, memory is assumed to be present on the wireless sensor node to hold (temporary) variables. Wireless sensor nodes require memory also to reside message queues for networking.

- *Power Supply Unit* of the sensor nodes generally consists of batteries where energy for operation of the wireless node is extracted from energy stored in chemical compounds. The power supply of a wireless sensor determines how much energy can be spent during the lifetime of the node.

Optionally, depending on the application requirements, a sensor node may have the following additional components:

- *Position finding system* provides physical location information of a sensor node. To interpret the sensor value, also the location of where a sensor reading was obtained must be known for some applications. Routing techniques may also need knowledge of the physical location of a sensor node. The positioning system may consists of a Global Positioning System (GPS) [14] module, which is a satellite navigation system, or a software module that implements the GPS-free localization algorithms, which provide location information through distributed calculations [29].

- *Power Generator* can be used in wireless sensor node to extracts energy from its environment. Heat, light, vibrations are converted to electrical currents, which power the node and optionally charge backup batteries. For instance, for outdoor applications, solar cells are used to generate power. Energy scavenging is the most preferred solution to the energy problem; however, the efficiency of most methods is still uncertain.

In certain environments energy might be plenty available whereas in others energy is truly scarce.

- *Mobilizer* is needed in cases where a sensor node has to move from one location to another. Although the advances developed by the distributed robotics and low power embedded systems enable attaching mobilizer to a sensor node, mobility requires extensive energy resources. A mobilizer also operates in a close interaction with the sensing unit and the processor to control the movement of the sensor node [29].

The characteristics of a wireless sensor network are determined by the characteristics of sink nodes and the characteristics of sensor nodes, which are determined by the application requirements. In what follows the characteristics of sink and sensor nodes are discussed:

Characteristics of Sinks

It is commonly agreed that a sink node is a powerful device with unconstrained energy supply and computational capacity. However, the following characteristics of sinks may critically influence the operations of communication in sensor networks.

- ***Number*** – Although the typical number of sinks is one, in most of the practical applications, an increased number of sinks provides more robust data gathering, and may help to increase the network lifetime and decrease the network delay. If only one sink is present, the destination for most of the data generated in the network is the same, whereas in case of multiple sinks, the destination sink may differ. Node to sink communication is more elaborated in Section 1.2.

- ***Mobility*** – During the network lifetime, the sink can be stationary or mobile. In some cases mobility inferred by the application, e.g. sinks are integrated with other mobile devices such as mobile phones carried by mobile users, in some others mobility ensures efficient data collection, the sinks move during the data gathering. To support sink mobility, it is crucial to provide means to reach the mobile sink node. Since frequent location updates from mobile sinks can generate excessive energy consumption of sensors, routing strategies handling sink mobility should provide efficient ways for the tracking of sinks in order to keep all sensor nodes updated for the future data reports.

- ***Presence*** – The sinks can be either continuously or partially present during the network lifetime. In the latter case, the routing protocol has to support the temporary lack of a sink. Instead of dropping messages in the absence of the sink, messages can be buffered at the source nodes or some other predefined locations (i.e. a set of sensors close to the sink) to send them to the sink when it is again reachable.

Characteristics of Sensor Nodes

The following characteristics of sensor nodes may vary for different WSN application; hence, they have influences on the operations of the network.

- ***Deployment*** – Sensor nodes can be deployed either in a deterministic or a random fashion. Generally indoor deployments (e.g., in a metro station, a school, etc.) require a deterministic approach rather than random. In these cases, the sensor nodes remain stable at their positions during the network lifetime. On the other hand, many applications assume that the deployment is done randomly, e.g., dropping the sensor nodes from a helicopter flying above a forest.

- ***Mobility*** – Sensor mobility can be also possible by attaching sensors to moving objects (e.g., animals) or by attaching a mobilizer device [52] to the sensor nodes. The mobility of sensor nodes changes the topology of the network. The presence of sensor mobility thus involves continues updating of the neighboring information and network structure (e.g., routing tree) for efficient message dissemination.

- ***Addressing*** – There are two fundamental goals in a WSN: (i) delivering queries sent by a sink to the sensors nodes which have the requested data (in case of query-driven reporting model as explained in Section 1.3), and (ii) returning a response including the requested data to the sink.

  – *Query Addressing*: A query-driven network employs a data-based (What is the average temperature in the sensor field?), or a region/location-based (What is the average temperature in the region surrounded by the circle having a radius r and centered at (x, y)?) addressing.

  – *Response Addressing*: The response with the data is either returned on the reverse path which the query traversed, or routed back in an ad-hoc manner or purely based on the location of the sink which has initiated the query.

1.2 Communication Patterns in WSNs

The communication of sensor devices is generally achieved by wireless RF (radio frequency) communication by means of the antennas, which emit and capture radio signals. Some other communication methods, such as infrared or microwave communication, are also possible. In this thesis, the WSNs under consideration use RF communication for networking of sensors. Sensors have small transmission range because of their low power radio. Due to the small communication range, messages are transmitted from a source node to a destination node using intermediate nodes in a hop-by-hop manner in the network. This results in *multi-hop* networking where sensor nodes relay each other's (i.e. neighbors') messages towards a destination in addition to their own data.

As we briefly mentioned in the previous section, there are two main types of communication patterns in a typical wireless sensor network: *Sink-to-Node* and *Node-to-Sink* communications. The most common form of node-to-sink communication pattern is called *convergecast* (many-to-one) where the sensor nodes report their data to a sink node. If there are multiple sinks and all the sinks must be informed about the data (e.g., event message), every source node has to send their messages to every sink in the network. This results in a communication pattern of *multicast* (opposite of convergecast – one-to-many) from a source node to every sink node in the network. The *multicast* communication pattern is also used for message dissemination (e.g., querying) from a sink to the sensor nodes. The other common communication patterns are *unicast* (one-to-one) and *local broadcast* (i.e. a node transmits data to all its neighboring nodes). The last two are employed when data is exchanged among the neighbors (i.e. local *node-to-node* communication pattern), for instance, for security key management and authentication, localization, or collaborative processing of data instead of sending raw data [95].

As it can be understood from the previous discussion on different communication patterns for different needs, designing communication protocols for WSNs is closely related with application requirements [141]. Therefore, it is impossible to design a single communication protocol that functions effectively and efficiently for all kinds of WSN applications. In the following, we describe the traditional WSN requirements that remain a continuous concern in designing networking protocols:

- ***Energy efficiency*** – Since sensor nodes are often powered by batteries and it is often difficult or even impossible to change or recharge their batteries, it is crucial to reduce power consumption of sensor nodes so that the lifetime of the sensors, as well as the whole network is prolonged. For this purpose, communication protocols running on the nodes need to ensure that the energy consumption is kept to a minimum by reducing the number of messages that are transmitted in the network since a sensor node expends maximum energy for data communication.
- ***Scalability*** – As the size of the sensor network grows with the wide availability of economically viable sensor nodes, scalability with respect to the number and density of nodes becomes very essential in WSN applications to prevent the decline of the network performance.
- ***Adaptability*** – In WSNs, network topology changes frequently due to the node failures, damages, additions, energy depletion, or channel fading. Thus, communication protocols designed for sensor networks should be adaptive to such network changes.

Functionalities and properties of a networking protocol are highly application specific. Besides the standard requirements discussed above, there are other important requirements, e.g. reliability, latency, fault-tolerance, etc., which will be discussed in Section 2.5. In the next section, we explain the data reporting models and the corresponding requirements of these models in wireless sensor networks.

1.3 Data Reporting Models in WSNs

The primary task of a wireless sensor node is to collect useful data by monitoring phenomena in the surrounding environment and transmit these data reports to the data sinks for analysis and processing. Data reporting in WSN depends on specific needs of the application and also on time sensitivity of the data collected [31]. Data reporting models can be categorized as time-driven, query-driven or event-driven models [158]. In the following, we explain the characteristics of these models:

- ***Time-Driven*** – This reporting method is the basic pattern for applications that require periodic data monitoring. In this model, to save energy, sensors can be sleeping or turned-off most of the time and periodically they wake up, switch on their sensors, sense the environment and transmit the sensed data to the sink in periodic intervals. Periodic data does not need to be transferred reliably since it has generally the same content as the previous reading.

- ***Query-Driven*** – A typical way of extracting data from a sensor network is to disseminate queries from sink nodes to sensor nodes, asking them to send data which has the properties specified in the queries. In this model, sensors only transmit data when it is explicitly requested by the sink. The sink may also send a query for some other purposes such as to specify or change the operation mode of a group of sensors (i.e. data sample rate etc.) or to update the system software running on the sensor nodes.

- ***Event-Driven*** – In this approach, whenever an event of interest occurs in the environment, e.g., the temperature rises above a certain threshold, the sensors report the data associated with that event to the sink. Usually events are rare. However, when an event occurs, a burst of packet is generated that needs to be delivered to the sink as quick and as reliably as possible.

For different applications, a combination of different data reporting models, which is called *hybrid model*, is also possible. In networks, where different data reporting models coexist, the routing protocol should change its operation depending on the importance of the information in data packets. For example, in a hybrid of time-driven and event-driven application, periodic sensor readings, which are collected to get an impression of the environment, have to give priority to event reports.

The routing protocol is highly influenced by the data reporting model in terms of energy consumption and route calculations [31]. While time-driven data reporting based applications may tolerate delay and loss of data, timely and reliable delivery of data may become very important concerns for query-driven and event-driven applications. Hence, both the query-driven and event-driven approaches are data-centric and well suitable for time critical applications. This thesis focuses on query-driven and event-driven data reporting patterns as we describe in the research question and the contributions of this thesis in the next sections.

1.4 Research Question

In view of the above communication and data reporting patterns, this thesis focuses on *Sink-to-Node* and *Node-to-Sink* communications in *query-driven* and *event-driven* wireless sensor networks, respectively. Traditional WSN requirements discussed in Section 1.2 are taken into account, as well as additional application specific requirements such as *adaptivity to mobility of sensors and sinks*. Mainly, we focus on query and data dissemination that require *timely* and *efficient* delivery of (query and/or data) messages in WSNs. We conceive the main research question as follows:

How can sink-to-node and node-to-sink communications be achieved in an efficient and effective manner in a multi-sink mobile wireless sensor network?

We approach the problem by taking into account the mobility of sinks and sensor nodes. The thesis addresses the question by following a bottom-up approach in three main scenarios with (i) static multiple sinks & static sensors, (ii) static multiple sinks & mobile sensors, and (iii) mobile multiple sinks & (mobile) sensors. We provide application examples of the above mentioned classes of scenarios in Section 2.4.

In the first scenario, we focus on the use of multiple sinks instead of a single sink in wireless sensor networks. WSNs can benefit from usage of multiple sinks. However, network

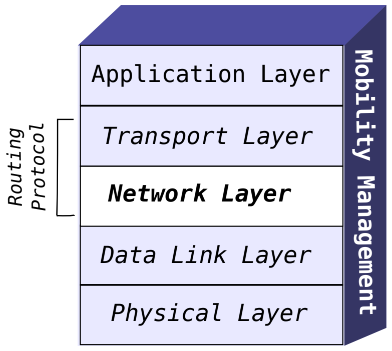


Figure 1.2: *Protocol stack for WSN requirements*

load balancing between sinks in the network and sensors in each cluster make multi-sink WSNs challenging. First, we study the load balancing in multi-sink WSNs where sinks and sensors are *static*.

Having explored the characteristics of multi-sink WSNs, we focus on protocols in the second part. We introduce a query dissemination protocol with region-based query addressing that utilizes special coverage area descriptions of sinks and sensors and also handles *sensor mobility* in a WSN with multiple *static* sinks.

In the third part, we explore a critical question: *how can multiple mobile sinks efficiently collect data from a mobile wireless sensor network?* We focus on a hybrid data reporting model of query- and event-driven approaches. We introduce an efficient data/query dissemination protocol which meets the traditional requirements of WSNs such as energy efficiency as well as timely and reliable data delivery. We also study fault tolerance for reliable data collection in multi-sink mobile WSNs.

Our hypothesis is that addressing dynamics of sink and sensor nodes in multi-sink deployments requires special attention calling for networking approaches that respond to specifics of applications. Moreover, all different mobility patterns (e.g., sink mobility, sensor mobility) have their special properties, so that each mobile device class needs its own approach.

1.4.1 Research Approach

The protocol stack used by sinks and sensor nodes is given in Figure 1.2 [30]. It is a more compressed version of the OSI (Open System Interconnection) model [179] of traditional communication networks. It is more compressed due to the fact that wireless sensor networks are application specific networks. In addition, the borders between layers are more flexible, in order to optimize the protocol stack for memory and energy consumption. As shown in the figure, the *mobility management* plane, which detects and registers the movement of sinks and sensor nodes [30], can cooperate with any layer of the protocol stack of WSN. Generally, these layers obtain, store and manage mobility information individually. From a communication perspective, mobility can be handled either by the medium access control (MAC) protocol [10] at the data link layer or by the routing protocol in the networking layer. Cross-layered approaches [32] are also proposed for mobility management. This thesis

Table 1.1: *Research Issues and Corresponding Contributions of the Thesis*

Research Issue	**Contribution**	**Chapter**	**Sink Mobility**	**Sensor Mobility**
Multi-sink partitioning	1-Load balancing between multiple sinks	3	–	–
Query dissemination in multi-sink WSN	2-Geocasting of queries	4	–	–
	3-Handling sensor mobility in geocasting	4	–	✓
Data dissemination in multi-sink WSN	4-Handling sink/source mobility efficiently	5	✓	✓
	5-Evaluation in highly mobile sensor networks	6	✓	✓

focuses on handling mobility in the *network layer*, which is implemented by the routing protocol.

1.5 Contributions

With regard to the previously mentioned research question and approach, we describe the main contributions of the thesis. Table 1.1 clarifies the relations between the research issues and our contributions.

✓ ***Contribution 1: Benefits and challenges of using multiple sinks in static wireless sensor networks:*** There are significant advantages of having *multiple sinks* in the network in terms of latency and energy consumption of information acquisition. The multi-sink partitioning of the network should be done by taking load balancing issues into account to acquire these benefits. We review the state of the art load balancing methods in wireless sensor networks. We present a mechanism for load balancing between sinks in the network and between sensors in each partition. We investigate the characteristics and problems of static multi-sink wireless sensor networks with extensive simulations. This work appeared in the following paper [4]:

– *A cross-layered communication protocol for load balancing in large scale multi-sink wireless sensor networks*, with T. Mutter, L. van Hoesel and P. Havinga, in Proceedings of the 9th International Symposium on Autonomous Decentralized Systems, ISADS 2009, pages 1-8, Athens, Greece, March 2009.

✓ ***Contribution 2: Design and evaluation of a query dissemination protocol for multi-sink wireless sensor networks:*** To enable sinks to efficiently route queries that are valid in particular regions of the deployment, we propose a set of algorithms that combine coverage area reporting and geographical routing of queries injected by sinks. Each sink constructs a routing tree and defines a coverage area and then geocast the queries in their coverage areas. Our geocasting protocol is designed for eliminating unnecessary query injections from sinks whose coverage areas do not intersect with the destination region. With this approach, we aim at decreasing energy consumption whereas meeting the requirements such as high query delivery ratio and low delivery delay. We study the case where sinks are static and sensor nodes may be mobile.

We evaluate the performance of our geocasting protocol with extensive simulations in both static and mobile scenarios and compare its performance with another well-known geocasting protocol. The work appeared in the following papers [11, 9]:

– *Combined Coverage Area Reporting and Geographical Routing in Wireless Sensor Actuator Networks for Cooperating with Unmanned Aerial Vehicles*, with L. van Hoesel and P. Havinga, in Proceedings of the 3rd ERCIM Workshop On eMobility, pages 43-54, Enschede, The Netherlands, May 2009.

– *Geo-casting of Queries Combined with Coverage Area Reporting for Wireless Sensor Networks*, with L. van Hoesel, A. Dilo, and P. Havinga, Ad Hoc Networks, Elsevier, Under Review, Submitted in July 2010, Revised and resubmitted in June 2011.

✓ ***Contribution 3: Handling mobility of sensors in the tree-based query dissemination protocol:*** To provide an up-to-date coverage area description to sinks, we focus on handling sensor node mobility in the network. We discuss what is the best method to handle mobility in tree-based routing of queries: (i) periodic global updates initiated by sinks or (ii) local updates triggered by mobile sensors. We propose a method to perform local updates in a tree-based network. With the extensive simulations we observe that local updates perform very well in terms of query delivery ratio, and also more energy efficient than global updating in networks having medium mobility rate and speed, independent of the size of the network. This contribution is submitted for publication in the following paper [7]:

– *On Mobility Management in Multi-sink Sensor Networks for Geocasting of Queries*, with A. Dilo, L. van Hoesel, and P. Havinga, Sensors, MDPI, Under Review, Submitted in May 2011.

✓ ***Contribution 4: Design and analysis of a data dissemination protocol for mobile multi-sink sensor networks:*** To achieve reliable data dissemination of events as well as the efficiency in handling the mobility of multiple sinks and event sources, we propose a virtual infrastructure and a data dissemination protocol, namely HexDD (Hexagonal cell-based Data Dissemination) exploiting this infrastructure. We analytically compare the communication cost and hot region traffic of the data dissemination with other approaches. Different parts of this work appeared in the following papers [1, 5]:

– *Data dissemination of emergency messages in mobile multi-sink wireless sensor networks*, with P. Havinga, in Proceedings of the 9th IFIP Annual Mediterranean Ad Hoc Networking Workshop, Med-Hoc-Net 2010, pages 1-8, Juan Les Pins, France, June 2010.

– *A fault-tolerant data dissemination based on Honeycomb Architecture for Mobile Multi-Sink wireless sensor networks*, with A. Dilo, and P. Havinga, in Proceedings of 6th International Conference on Intelligent Sensors, Sensor Networks and Information Processing, ISSNIP 2010, pages 97-102, Brisbane, Australia, December 2010.

✓ ***Contribution 5: Evaluation of the data dissemination protocol for different wireless sensor network applications:*** We focus on the performance evaluation of data dissemination in two different classes of mobile wireless sensor networks with *mobile* sinks: (i) Emergency Response Application – *mostly static*, which contains scenarios in which most of the sensors are static and some sensors are attached to people or vehicles such as firefighters or unmanned aerial vehicles moving at low or medium velocities, (ii) Vehicular Sensor Network Application – *highly mobile*, which contains scenarios in which many sensors are attached to devices that move at high velocities such as cars. We compare the performance of HexDD protocol with other application-specific protocols in terms of data delivery ratio, latency and energy efficiency. We investigate the effects of number of sinks and speed of mobile sinks on the performance of the dissemination protocol. This work is submitted for publication in the following papers [6, 8]:

- *A virtual infrastructure based on honeycomb tessellation for data dissemination in multi-sink mobile wireless sensor networks*, with A. Dilo, and P. Havinga, EURASIP Journal on Wireless Communications and Networking, Hindawi, Under Review, Submitted in April 2011.
- *Infrastructure Assisted Data Dissemination for Vehicular Sensor Networks in Metropolitan Areas*, with R. S. Schwartz, A. Dilo, H. Scholten, and P. Havinga, Roadside Networks for Vehicular Communications: Architectures, Applications and Test Fields, IGI-Global, Under Review, Submitted in April 2011.

1.6 Organization of the Thesis

In the next chapter (Chapter 2), we provide the reader with an overview of multiple sinks and mobility in wireless sensor networks including related applications and their challenges [3, 2]. The rest of the chapters are blocked into 3 groups as shown in Figure 1.3:

(i) Chapter 3 focuses on load balancing between sinks and sensors in static multi-sink wireless sensor networks which corresponds to *Contribution 1*.

(ii) Chapter 4 describes geocasting of queries in multi-sink wireless sensor network (*Contribution 2*) and analyzes different approaches (i.e. global updating and local updating) for geocast structure maintenance to handle sensor mobility (*Contribution 3*).

(iii) Chapter 5 introduces and analytically evaluates our virtual infrastructure based data dissemination protocol, namely HexDD, in mobile multi-sink wireless sensor networks (*Contribution 4*). We evaluate the performance of HexDD by comparing with two other virtual infrastructure based data dissemination protocols designed for WSNs in Chapter 6. In addition, in Chapter 6, HexDD protocol is tested in a highly mobile scenario, Vehicular Sensor Network, which corresponds to *Contribution 5*.

We conclude this thesis in Chapter 7 summarizing the key results and highlighting the possible future research directions for the problems and solutions presented in the thesis.

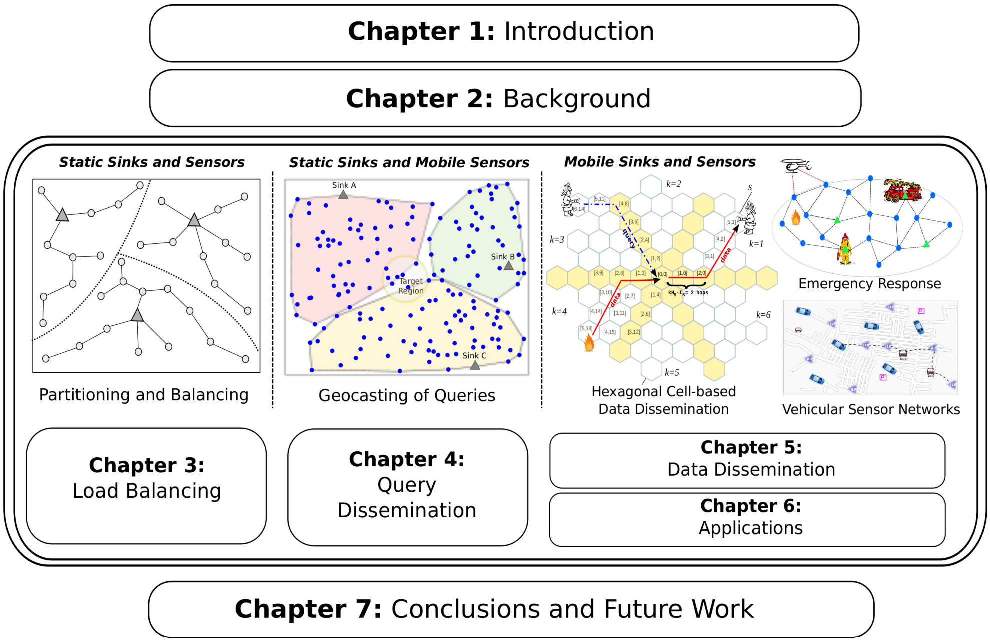

Chapter 1: Introduction
Chapter 2: Background
Static Sinks and Sensors
Static Sinks and Mobile Sensors
Mobile Sinks and Sensors
Sink A
Sink B
Sink C
Target Region
Partitioning and Balancing
Geocasting of Queries
Hexagonal Cell-based Data Dissemination
Emergency Response
Vehicular Sensor Networks
Chapter 3: Load Balancing
Chapter 4: Query Dissemination
Chapter 5: Data Dissemination
Chapter 6: Applications
Chapter 7: Conclusions and Future Work

Figure 1.3: *Organization of the thesis*

CHAPTER II
Background

The focus of this thesis is on efficient query and data dissemination in mobile wireless sensor networks with multiple sinks. This chapter introduces the existing approaches on the general concepts studied in this thesis (i.e. multiple sinks and mobility), while the forthcoming chapters present the existing work specific to the topic studied. Firstly, we discuss the need and usage of multiple sinks in WSNs. Secondly, we highlight the main characteristics of mobility of different entities in WSNs and elaborate how mobility helps to alleviate some of the traditional problems associated with static sensor networks. We also present different mobility models together with a survey of exiting work on these models. Next, we describe well-known applications of WSNs to illustrate how this technology is integrated with our everyday life. We go on to give a detailed description of four applications that are particularly relevant to this thesis as they have provided the motivating factors behind the design decisions we have made in the forthcoming chapters. Finally, based on the requirements introduced by the applications relevant to this thesis, we highlight the main properties that should be present in the protocols designed for such applications.

2.1 Multiple Sinks in Wireless Sensor Networks

Data gathering is a fundamental task of a WSN [102]. It aims to collect sensor readings from sensory field at a predefined *sink* for analysis and processing. The well-known problem of a *single-sink* WSN is the *uneven energy depletion phenomenon*. Sensors near the sink deplete their battery power faster than those far apart due to their heavy overhead of relaying messages. Non-uniform energy consumption causes degraded network performance and shortens network lifetime. If all sensors around the sink run out of energy, the sink will be isolated from the network, then the entire network fails. There are significant advantages of having *multiple sinks* in the network in terms of latency and energy consumption of information acquisition. First of all, having multiple sinks in the network alleviates the unbalanced energy consumption among sensor nodes since the data transmission load is shared among all the sinks. In addition, multi-sink networks can remarkably reduce the mean distance (and also hop count) between sensor nodes and sinks, basically resulting in energy saving and lower latency. Finally, multiple sink deployment avoids the single bottleneck problem since in case of disconnection of one particular sink from the network, sensors can still transmit their data towards other sinks, and the network continues to function.

Depending on the application requirements, a WSN with multiple sinks can be divided into sub-networks (i.e. clusters), each of which is composed of a single sink. In each of these sub-networks, data sources report their reading to the sink of the cluster, and sensor nodes in the cluster relay messages sent by the sources towards the sink. In order to form such a WSN, there are two typical cases for sink deployment:

(i) First case is deciding explicitly where to deploy sinks inside the sensor field, which is called *Sink Location Problem*. This is a typical '*facility location*' problem: given a set of 'sink nodes' (i.e. facilities) and a set of 'sensor nodes' (i.e. customers) to be served from these sink nodes, where to deploy the sinks (i.e. facilities), and which sink should serve which sensor node (i.e. customer), so as to minimize the total '*serving cost*' (e.g., the overall energy consumption)?

(ii) Second case is randomly deploying a predefined number of sinks inside the sensor field. In this case, the main problem is finding efficient routes from sensors to the sinks, as well as finding the best partitioning (i.e. clustering) of the network area into regions corresponding to the sinks.

The number and the exact locations of sink nodes directly affect the network lifetime of a WSN. Depending on the design objective, there might be several approaches for finding the number and the location of the sink nodes [28, 127]. The *sink positioning* problem is typically defined as finding the optimal layout for a known number of sinks in the sensor field to maximize a performance metric, such as total communication energy and throughput or area coverage. This is often solved by efficiently clustering sensors inside the sensor field [113, 133, 134]. The center of mass of sensors within a cluster would give the location of a sink node. Another approach is *minimization of the number of sink nodes for a predefined minimum operation period*. In order to solve this problem, the sensor network lifetime for any number and positioning of sinks has to be calculated [127]. Then, the minimum number of sinks providing a network lifetime that exceeds the predefined limiting constraint will be selected as the solution. A similar problem is *minimization of the number of sink nodes while maximizing the network lifetime* when there is neither prior knowledge on the number of sinks nor the lifetime constraint. The above approaches for optimizing the placement of sinks in multi-hop wireless sensor networks are NP-hard problems [64].

Solutions to the sink location problem require deterministic deployment where sink nodes are placed deliberately in the sensor field. On the other hand, in most of the large-scale WSN applications the sensor and sink deployment is done in a random fashion (e.g. by dropping the sensor and sink nodes from a low-flying airplane over a vast area such as a forest). Non-deterministic deployment is suitable for both large-scale applications and hostile environments. In this thesis, we consider sensor network applications (see Section 2.4) where we have randomly deployed sensors and sinks inside the sensor field. In such applications, the main concern is generally finding the efficient clusters and/or efficient routes from sensors to sinks. We explain the details of related works on these topics in Chapter 3, where we survey multi-sink clustering approaches in static WSNs and investigate the performance of different multi-sink clustering strategies and routing methods considering different metrics (e.g. energy level of nodes).

2.2 Mobility in Wireless Sensor Networks

Recent research efforts [35, 71, 82, 88, 90, 161] have showed that the use of mobile elements can enhance connectivity and lifetime of WSNs. In many deployment scenarios, mobile entities already exist in the deployment area, such as firemen in an emergency response application, and buses in a traffic monitoring application (see Section 2.4 for details on these applications). The mobile nodes, which are capable of communicating with other nodes in the network, can address the connectivity problem by carrying information between isolated (disconnected) parts of WSNs. Since mobility has been proposed as another way for reducing the communication distance between sensors and sinks in the literature, network lifetime can be improved with mobile devices by reducing multi-hop communication. Another important problem of static deployments is the bottleneck problem, which appears on the nodes close to the sink. As all the data is forwarded towards the sink, the average load on a sensor node increases with decreasing distance between the node and the sink [114]. Mobility also helps to solve this problem by deploying mobile sensor nodes and sinks in the network. Wang et al. discuss in [165] how mobile elements improve the network lifetime. Figure 2.1 compares

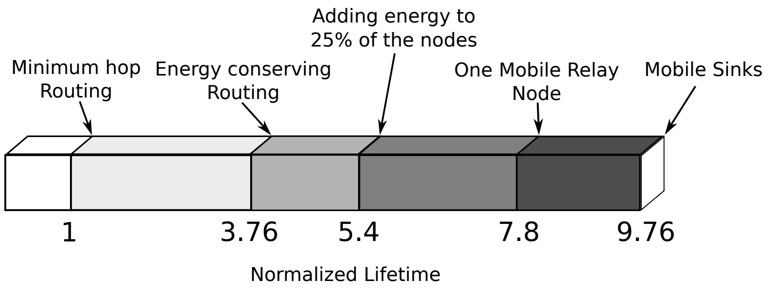


Figure 2.1: *Comparing network lifetimes of different approaches [165]*

the network lifetime of different approaches. In the figure, the lifetime of the non-energy aware minimum hop routing in a static network is normalized to 1. Energy conserving routing, which gives the upper bound for a static WSN, improves the network lifetime nearly four times over minimum hop routing. While using static energy-provisioning scheme and adding four times more energy to randomly selected 25% of the static nodes, the lifetime can only be extended by 40%. On the other hand, by adding only one mobile relay[†] node, the lifetime of energy conserving routing can be doubled. Using mobile sinks gives a lifetime improvement which is even better than a mobile relay node.

In wireless sensor networks, mobility can appear in three main forms [87]: *mobility of the sensor nodes* that sense the environment and transmit the related information, *mobility of data sinks* that gather the information from the network and forward data to the application, and *mobility of the observed event*.

- **Sensor Node Mobility:** Figure 2.2(a) illustrates sensor node mobility in a livestock surveillance application (e.g. sensor nodes attached to cattle). The mobility of sensors can influence protocols at the networking layer. The network has to reorganize itself frequently enough to be able to function correctly. There are trade-offs between the frequency and speed of node movement on the one hand and the energy required to maintain a desired level of functionality in the network on the other hand [87].

- **Sink Mobility:** Since the sinks are requesters of the data, the mobility of sinks may occur due to the mobility of end-users carrying sink nodes in the network. For example, in a disaster response application, a fireman requests event related data from sensor nodes while he is moving inside the network as illustrated Figure 2.2(b). Generally speaking, sink mobility is offered as a solution for uneven energy depletion phenomenon described in Section 2.1. The sink mobility also has a great influence on protocols at the networking layer. The network, possibly with the assistance of the mobile sink, must make provisions that the requested data actually follows and reaches the sink despite its movements.

- **Event Mobility:** The cause of the events or the objects to be tracked can be mobile in applications like event detection and in particular tracking scenarios. The mobile event scenario is described by Figure 2.2(c), where the task is to detect a moving tank in a border protection application and to observe it as it moves around. Since the location of the event changes over time, the sensor nodes sensing and reporting the event (i.e. source nodes) also change over time.

The mobility of event [172] is highly application dependent, meaning that it can not be controlled (mostly also unpredictable) by the sensor network. Generally, event mobility appears in tracking applications such as animal tracking or military applications (e.g. tracking enemies) which are not under consideration of this thesis. In this thesis, we focus on mobility of sensor and sink nodes. Without losing the generality, mobility of a (sink or sensor) node can be classified as follows:

[†] A mobile relay node, which stays within a two-hop radius of the sink, takes over the tasks of multiple bottleneck nodes close to the sink during different network time periods.

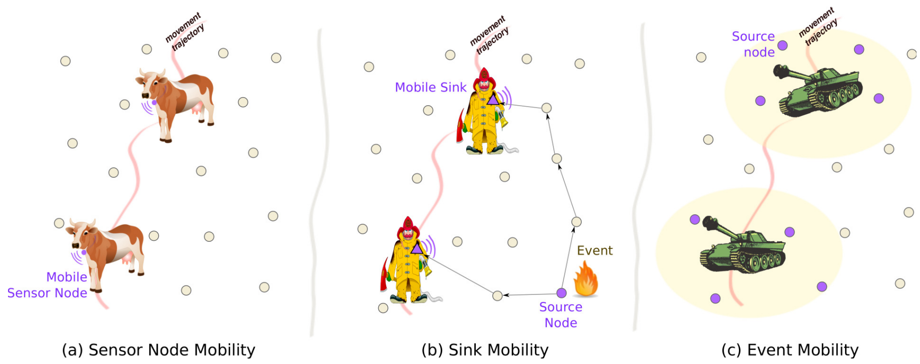


Figure 2.2: *(a) Sensor nodes attached to cows are moving inside the sensor field, (b) Fireman having a sink collects data from sensors while it is moving, and (c) Area of sensor nodes detecting an event – a tank – that moves through the network along with the event source*

- **Uncontrolled Mobility:** A node moves in a random fashion. The mobility of a node can be considered as uncontrollable (or random) if the motion characteristics (i.e. direction, trajectory) of the node is not related with or determined by the data routing requirements. It is regulated according to the primary purpose of the user (i.e. people, vehicle) carrying the node in the sensor field. For instance, the purpose of a fireman carrying a sensor and/or sink node is eliminating fire in the fire field while getting information about the situation for the sensor network deployed in the field, thus his movement is designated by his primary task. Therefore, the sensor/sink mobility is considered as uncontrollable in such a scenario. The main problem in this kind of scenarios is how to deliver data from a source node to sinks when the intermediary sensor nodes and/or sinks are randomly moving in the network.

- **Predictable Mobility:** Predictable mobility refers to the case when the motion is known but cannot be changed. However, this knowledge can be exploited to route data. Predictable sink and sensor mobility can improve energy efficiency of data transmission [115] by combining data relaying with predictable mobility. Sensors can predict the time of data transfer by utilizing the trajectory of the mobile node. Based on the predicted data transfer time, the sensors become active at the time of data transfer, otherwise they sleep until the time of data transfer comes to save energy. A representative example of predictable mobility is the vehicular mobility such as public transportation (i.e. train, bus). Such vehicles can act as mobile sinks in wide area WSNs for applications such as pollution monitoring.

- **Controllable Mobility:** The mobility of the node can be controlled by a user. Controllable mobility, like sensors mounted on a robot [52], can be used as means for improving network connectivity and data dissemination tasks. The controlled mobility of sensor nodes is generally used to achieve optimal deployment or to cover holes in connectivity or sensing coverage. Sink mobility is controlled usually for the purposes of avoiding hot spots around the sink and distributing energy consumption throughout the network evenly or enabling single-hop communication between the sink and the

Table 2.1: *Motivation of Different Types of Mobility in Wireless Sensor Networks*

Mobility	Sink	Ref.	Sensor	Ref.
Uncontrolled	• Inherent in the scenario	[3]	• Inherent in the scenario	[3, 71]
	• Avoid hot spots around sink			
Predictable	• Avoid hot spots around sink		• Utilize scheduled transmission	[61]
	• Achieve single-hop communication (at the cost of latency)	[39]		
	• Utilize scheduled transmission	[161]		
Controllable	• Avoid hot spots around sink		• Achieve optimal deployment	[82]
	• Cover holes in connectivity	[88]	• Cover holes in connectivity	[164]
	• Increase network lifetime	[35, 90, 103]	• Increase network lifetime	[86]
	• Achieve single-hop communication (at the cost of latency)	[65]		

source nodes to avoid long-range communication. In these approaches of exploiting controlled mobility of sink, the problem is mostly about finding the optimal motion of the mobile sink to balance the energy consumption in the network or scheduling the mobile sink in real time to visit source nodes such that no sensor buffer overflow occurs and the data loss is avoided [65].

Table 2.1 shows the motivation of different types of sink and sensor mobility with the related works done in these fields. Indeed, no matter for what purpose or through what means (i.e. random or not) if sink mobility is present in the network, it always helps to avoid bottlenecks around the sink. Controlled and predictable mobility of sink and sensor nodes are well-studied topics in the literature as shown in the table. In general, controlled and predictable mobility are incorporated in a WSN and are exploited for improving the network performance. On the other hand, uncontrolled mobility is highly application dependent and is inherent in the scenario. Therefore, it requires special cautions to achieve efficient networking in a WSN. In this thesis, we focus on the uncontrolled (random) mobility of sinks and sensors. In Chapter 4, we consider a hybrid architecture combining a fixed infrastructure network with mobile sensor nodes. In Chapters 5 and 6, we go one step forward by considering mobile sinks in both moderately and highly mobile applications.

In what follows we first provide a brief list of general examples of WSN applications. We then describe some specific applications that are particularly relevant to the work presented in this thesis.

2.3 Applications of Wireless Sensor Networks

Based on recent technological advances in wireless communication, low-power microelectronics integration and miniaturization, the manufacturing of a large number of low cost wireless sensors became technically and economically feasible. Hundreds or thousands of these constrained devices with relatively small memory resource, restricted computation capability, short range wireless transmitter-receiver and limited built-in battery, can potentially be networked as a wireless sensor network (WSN) for many applications that require unattended, long-term operations.

There are a number of different application fields for sensor networks ranging from traditional data gathering (e.g. environmental monitoring) to more complex applications (e.g.

emergency response). Consequently, WSNs have emerged as a promising technology with various applications:

- **Environmental monitoring:** Environmental monitoring is one of the earliest applications of sensor networks. Environmental sensor networks are often large scale, static, non-dense, and are deployed in harsh and unattended environments. Networked sensors enable long-term data collection such as temperature, humidity, pollution, agricultural data, etc. at scales or resolutions that are difficult to obtain by other means. Monitoring Great Duck Island [117] in USA for the distribution and abundance of plants and animals, monitoring the Great Barrier Reef in Australia [41] for coral bleaching, and the impact of temperature on aquatic life and pollution, irrigation management in agriculture and landscaping by monitoring soil moisture [12, 140], detecting foreign chemical agents in the air [56] and the water are some examples of environmental monitoring applications of WSNs.

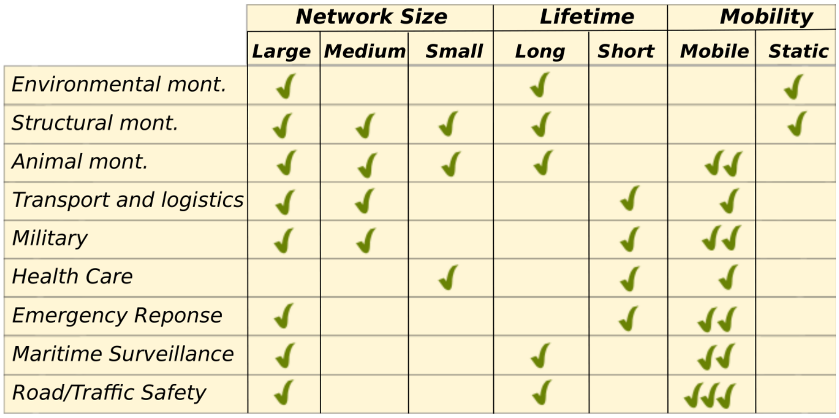
- **Structural monitoring:** Wireless sensor networks are used for structural health monitoring of buildings, bridges, tunnels and other structures to estimate the state of structural health, or detecting the changes in structure that influence its performance [91], and also used to monitor natural risks such as landslide and rockfall areas that create hazardous situations. Medium to large numbers of wireless nodes are deployed inside or outside of structures and remain static. The GENESI project [20] founded by EU focuses on monitoring a structure while being build (e.g. tunnels), and continuous health monitoring (e.g. a bridge).

- **Animal monitoring:** Typical applications of animal tracking is monitoring animals for studying their behaviors, and locating or confining them in pastoral regions or in the wild. Monitoring of a typical farm environment [136], in particular cattle monitoring, and wildlife tracking [84] in the ecological system are the major examples of such applications. Different from the environmental monitoring applications, animal monitoring introduces mobility within the WSNs.

- **Transport and logistics:** For transport and logistics, goods can be monitored while they are in warehouses and when they are in transit. Generally, the goal of transport and logistics applications of WSNs is to monitor the storage conditions of products (e.g. flower warehouse monitoring [67]), to verify the loads, and to localize the goods and items at production side or distribution stores (e.g. Collaborative Business Items (CoBIs) project [13]). The sensor network deployments for transport and logistics applications form a hybrid network consisting of both static sensors (e.g. placed in warehouse) and mobile sensors (e.g. attached to rolling containers or carts).

- **Military:** Usage of WSNs in military applications helps to achieve effective battlefield situational awareness [100], theater control, enemy military reconnaissance, intrusion detection [96], target (enemy) tracking, war damage assessment, and nuclear/biological/chemical attack detection. The scale and the mobility type of military applications are very much dependent on the target application. For instance, intrusion detection comprises event mobility whereas battlefield situational awareness includes mobility of sensors attached to soldiers and tanks.

Table 2.2: *Overview of WSNs applications (number of tick in Mobile column reflects the mobility rate)*



	Network Size			Lifetime		Mobility	
	Large	Medium	Small	Long	Short	Mobile	Static
Environmental mont.	✓			✓			✓
Structural mont.	✓	✓	✓	✓			✓
Animal mont.	✓	✓	✓	✓		✓✓	
Transport and logistics	✓	✓			✓	✓	
Military	✓	✓			✓	✓✓	
Health Care			✓		✓	✓	
Emergency Reponse	✓				✓	✓✓	
Maritime Surveillance	✓			✓		✓✓	
Road/Traffic Safety	✓			✓		✓✓✓	

- **Health Care:** Health care applications use WSNs for acquiring physiological and behavioral data from patients for diagnosis, monitoring, or chronic disease management. Sometimes the data collected by the body area network is augmented with sensor readings from training objects or environmental sensors in home/hospital environment. Examples of this class include personal health care monitoring with body area networks [119], and tracking doctors and patients inside hospitals [74]. Since the sensors in health care application are physically grouped to form a body area network, it has a different mobility model, called group mobility [135].

The list of applications is certainly extendible since the number of application areas of sensor networks is continuously growing within the research and industrial communities. Table 2.2 gives an overview of the WSN applications by highlighting high-level features such as scale, network lifetime, mobility rate. As one can see from the table, most of the WSN applications include mobility. The rate of the mobility (i.e. number of mobile sensors per total number of sensors) may vary from weakly mobile to highly mobile for different applications. Generally, mobility model of the applications shown in the table is in the category of uncontrollable mobility.

2.4 Applications relevant to this thesis

In this section, we describe four applications that are of particular relevance to this thesis.

2.4.1 Monitoring rainforests

Rainforests are globally significant ecological systems. Scientists are still working to understand these complex systems and their impact on climate change. Rainforest monitoring is a typical environmental monitoring in a harsh environment. Hundreds of sensors are needed to measure temperature, water vapor, and solar radiation across hill slopes, essentially taking the vital signs of the rainforest. The information collected by the WSN is very valuable for scientists, who want to improve their understanding of this ecosystem. The deployment of a WSN at a rainforest allow the various areas of the forest to be monitored at high spatial

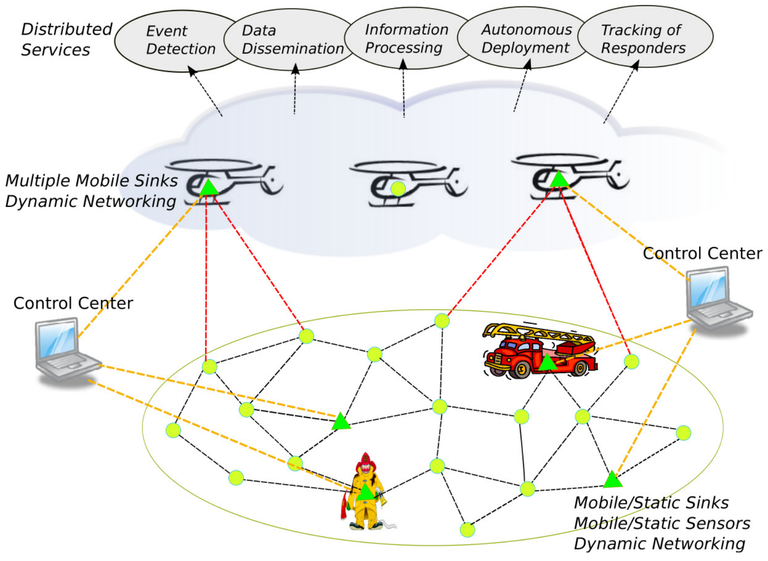


Figure 2.3: *WSN and communication structure in AWARE project*

and temporal resolutions. In order to deploy and develop an efficient WSNs in a rainforest, data collection protocols are required to enable *long-term*, robust and reliable performance. Generally, WSNs deployed in rainforests are *static*, *large-scale*, *multi-hop*, *multi-sink* networks. The network may include mobile entities if habitants (i.e. animals) of the rainforest are also monitored. Even though there is no mobile node in the network, the network is very dynamic since wireless links of the network are highly dynamic due to the frequent changes of weather conditions such as fog and rain. Moreover, the rainforest foliage has an effect on the link quality of longer hops [166], meaning shorter hops should be chosen.

Routing tree approach is considered as an efficient alternative to deliver data to the sink for such static applications. In this approach, the sensor nodes organize themselves in a routing tree rooted at the sink. Many of these scenarios often involve several sink nodes. As a starting point in Chapter 3, we consider a static network and focus on load balancing between sinks and inside each sink's routing tree by taking shortest path routing as the basis.

2.4.2 Emergency Response

Knowing how to both monitor and deal with a large number of catastrophic conditions is key to emergency response scenarios [111]. Multiple entities have to be optimally coordinated and numerous resources must be allocated efficiently, creating a very challenging activity. The emergency response scenarios are real-time applications, meaning that decisions have to be made in a timely manner during emergency response situations. Sensor networks offer a very good solution for emergency situation monitoring and management by capturing, processing, and communicating critical data for use by first responders (e.g., police, firemen, paramedics, etc.).

The Aware platform is developed as part of the AWARE EU Project [3, 19] and focuses on disaster management and civil security scenarios. It has been established to study the potential collaboration of a ground wireless sensor network with unmanned aerial vehicles (UAVs), e.g. autonomous helicopters. In this scenario, it is essential to establish an efficient communication and cooperation platform that is able to self-organize, adapt to changes and provide support in case of emergency situations. The architecture of the Aware platform comprises a number of heterogeneous sub-systems, which are described in relation to the global architecture in Figure 2.3. The Aware platform consists of a wireless sensor network with both static and mobile nodes, unmanned aerial vehicles (UAVs) and mobile devices carried by people which act as data sinks and/or sensor nodes.

The primary application of the project is a fire fighting scenario. In case of fire, sensor nodes are dynamically (randomly) deployed by UAVs into the area of interest (e.g. rainforest) and start measuring environmental characteristics such as temperature, humidity and gas level. Every sensor node is capable of precisely measuring its position via an embedded GPS receiver or an intelligent localization method. Based on the analysis of the sensor readings, the mission coordinator can rate the scale and the spread of fire, and eventually the location of fire-fighters and fire-trucks, and coordinate the situation effectively.

We have two key system layers of abstraction as shown in Figure 2.3: the *sensor and dynamic networking layer*, and the *distributed services layer*. The sensor and networking layer contains the sensor and the network protocols, which allow messages to be forwarded through multiple sensors taking into account the mobility of nodes and the dynamic change of topology. The described scenario involves two rather different communication networks: (i) the ground sensor network with fixed sinks, but mobile sensors carried by first responders (studied in Chapter 4), and (ii) the ground sensor networks with mobile sensors plus mobile sinks attached to UAVs or fire-trucks (studied in Chapter 5 and 6.1). The mobility of both sensors and sinks is considered as *uncontrollable* from networking point of view. In this application, there is a strong need for reliable and fast communication which can adapt itself to topology changes due to mobility of sinks and sensors.

2.4.3 Maritime Surveillance

Indeed, the characteristics of the WSN formed in maritime surveillance scenario are very similar to the characteristics of emergency response scenario on the sea. The key mission of any coastal or harbor surveillance system is to provide decision makers and first responders (e.g. coastguards) with full situation awareness of the coverage area. One of the most efficient coverage for large coastlines and harbors can be achieved through a wireless sensor network. Sensors are generally positioned in strategic locations to provide detection and classification of all cooperative and non-cooperative targets. The information collected from the WSN is used for protecting assets and supporting search and rescue missions.

The RECONSURVE project [21] offers a harbor surveillance system that is targeted against both surface and underwater threats. The sensor types that were chosen for a total surveillance capability are acoustic sensors, radars, optical sensors, and an Automatic Identification System (AIS). A network, which can detect both underwater and surface activities, is deployed closed to the harbor entrance points or sea borders by making use of buoys as illustrated in Figure 2.4. The sensor buoys report data to a control center only if an event of interest occurs. Usually, events are rare. Yet, when an event occurs, a burst of packets is often generated that needs to be transmitted reliably, and usually in real-time, to the control center.

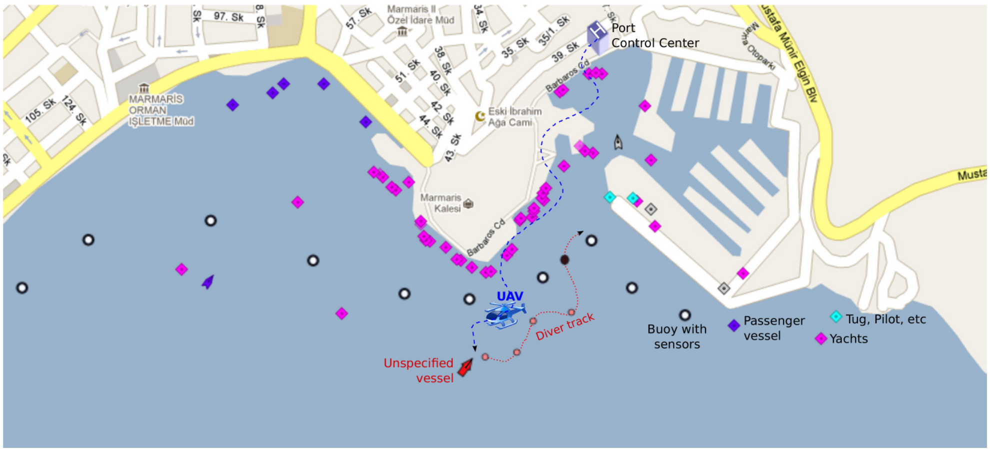


Figure 2.4: *Harbor surveillance scenario with an unspecified vessel and a plunger trying to reach Marmaris Harbor at Aegean Sea*

The success of the network depends on the efficient dissemination of the event. Generally, buoys are static, forming a fixed infrastructure network. The mobile entities of the network are events and responders which are both uncontrollable form networking point of view. On the other hand, UAVs can also be used in this application to patrol in the mission area, and to collect data for providing further details about the potential threat.

2.4.4 Situational Awareness on the Roads

Vehicular sensor networks have attracted people's attention in recent years with the vision that it can provide crucial information, such as traffic conditions (i.e. collision or traffic congestion around city center) to interested parties. Vehicles together with roadside sensor nodes form a hybrid network that can serve many applications, such as traffic monitoring and control, environmental monitoring, and safety warning. Many stationary sensor nodes are deployed on the side of the road that can detect, measure and record a certain aspect of the traffic pattern, such as the speed of vehicles in its range, or some other environmental conditions if they have the corresponding sensors. Nodes, similar to the stationary ones, are placed also in the cars that can communicate with the stationary roadside nodes and gather data from these roadside nodes to display data (related with traffic conditions, for example) on the computer in the car. Communicating cars and roadside infrastructure collectively forms a wireless sensor network. Cars can also communicate with each other, but prefer roadside units for long haul region-to-region data transfers.

Such a network can be considered as highly mobile since most of the sensors are carried by vehicles. Vehicles moving in the city are the mobile sensor nodes in the network. They send the information collected by their possibly many sensors to the static roadside nodes in the network. They also request data from the network, thus act also as sink nodes. The data dissemination protocol introduced in Chapter 5 is extensively evaluated in this scenario in Chapter 6.2.

2.5 Essential characteristics for WSN protocols

Based on the requirements of the applications described in the previous section, we now state some of the challenges – in addition to this list of traditional WSN requirements given in Section 1.2 – that should be addressed in protocols designed for such applications:

- ***Dynamics of the network*** – A major increase in the overall degree of mobility or dynamics can be observed from the applications discussed in the previous section. Contrary to traditional applications of WSNs, mobility becomes necessary for today's more complex applications of WSNs. Mobility thus becomes an *intrinsic system property*, which needs to be considered even from the protocol design phase [118]. Although mobility turns out to be a means of enhancing network performance as discussed in the previous section, routing of messages from or to moving nodes while also intermediate nodes are moving is more challenging since a continuous route establishment is needed to achieve route stability between source and destination nodes. The performance of the protocols designed for mobile WSN applications can be evaluated by measuring the data delivery ratio with varying mobility speeds and rates.

- ***Fault tolerance*** – Sensor nodes are prone to failures due to harsh environmental conditions. Sometimes a part of the network does not function anymore due to failure of a group of sensors which results in routing holes. The protocol design for such networks should be fault-tolerant and able to handle or recover from holes in the network. A protocol can be categorized as fault-tolerant if it can deliver high portion of the data to sinks, despite the size and the number of routing holes in the network.

- ***Real-time delivery*** – Some applications such as emergency response require that the messages must be delivered within a specified time, otherwise the message becomes useless or the importance of its information content decreases after a time bound. Therefore, one of the main objective that has to be considered in the protocol design is to control or minimize the network delay. The performance of the protocols designed for real-time applications can be evaluated by measuring delivery ratio with time constraints.

In each of the forthcoming chapters, the design decisions have been made by taking into account these requirements that are relevant to the specific application, which motivated the work studied in that chapter.

2.6 Conclusion

Although wireless sensor networks are one of the most promising technologies of the 21st century – with potential applications in virtually all areas of activity, ranging from the personal area to the global environment – a considerable number of challenges, which are discussed above, has still to be addressed in order to make WSNs a day-to-day reality. One of the most crucial aspects is mobility. Many applications require sensor mobility, and/or sink mobility, to be more effective. In next chapter, as a starting point we study on a static multi-sink WSN to understand the limitations of static deployments of sinks and sensors in the network. Without integrating mobility in WSNs and providing efficient mobility handling mechanisms, the application areas of WSNs will be highly restricted. In this thesis we propose and evaluate a set of protocols for an effective support of mobility in multi-sink WSNs by taking into account essential characteristics discussed in Section 2.5.

CHAPTER III *

Load Balancing in Multi-Sink Wireless Sensor Networks

One of the fundamental operations in sensor networks is convergecast, which refers to the communication pattern where data is collected from a set of sensor nodes and forwarded towards a common end-point gateway, namely sink node, in the network. In this chapter we study the case where there is more than one sink in the network. This work is the starting point of the thesis providing and understanding of the advantages and problems of multi-sink deployments in WSNs. In case of multiple sinks within the network, the total load of the network has to be balanced among these sinks and between sensors reporting to a sink to maximize the network functionality. We analyze different routing strategies that are used to achieve this goal and investigate the characteristics of networking in static multi-sink WSNs. We first present a cross-layered communication strategy for data collection in multi-sink WSNs. It basically combines a network wide inter-cluster load balancing technique with a metric-based intra-cluster shortest path routing. In this approach sinks collect information from nodes in the network about initial cluster sizes, then analyze and distribute this information back into the sensor nodes. Sensor nodes use this global information in combination with local information about the one-hop network neighborhood to build routing trees rooted at sinks. We explore the effects of different routing metrics on the networking performance with simulations. The performance evaluation of the presented techniques show the relationship between networking performance and global/local load balancing in static multi-sink WSNs.

*This chapter is an extension of the paper entitled "A cross-layered communication protocol for load balancing in large scale multi-sink wireless sensor networks" published in the Proceedings of the 9th International Symposium on Autonomous Decentralized Systems, March 2009 [4].

3.1 Introduction

Most of the sensor networks have one sink, but some may have multiple sinks. Having multiple sinks in the network gives advantages such as energy efficiency, reliability and alleviation of the hotspot problem which are discussed in Chapter 2.1. On the other side it adds additional requirements. The main concern in a multi-sink deployment is balancing the network load by partitioning the network between sinks. It is also important to balance the load between sensor nodes in each partition.

The aim of this chapter is investigating the advantages and problems of multi-sink WSNs as a baseline. We also present a routing strategy, namely Partition-based Network Load Balanced routing protocol (P-NLB), which utilizes the existence of multiple sinks in large scale WSNs. The protocol extensively uses the cross-layer information from the MAC protocol at the data link layer. Sensor nodes use this information to obtain a local view of the network neighborhood. Since an application may have different targets, such as long network lifetime, low latency or high data throughput, P-NLB is designed to be capable of dealing with these different targets. It uses a network-wide distributed inter-cluster load balancing technique in combination with a metric-based intra-cluster shortest path routing. In this two-level approach, sinks collect information from nodes in the network about cluster sizes and distribute this information back into the network. Sensor nodes use this global information in combination with local information about the one-hop network neighborhood to build a routing tree. In the setup phase of the network it is detected whether the network should function in load balancing routing mode or the basic metric-based routing mode. The network topology is the key factor in that decision. If the protocol detects that the cardinalities of the initial clusters in the network are not equal, it will activate the load balancing routing mode in order to restore the balance. Otherwise, inter-cluster load balancing is not necessary and the basic routing is activated. Both routing modes feature a metric-based routing tree building mechanism, which nodes use to follow a certain routing strategy, e.g. avoiding congested or nearly depleted nodes on the routing path. Different metrics are defined, such as buffer occupancy, remaining energy, number of child nodes, and number of dependent nodes of sensors, for building balanced routing trees.

The remainder of this chapter is organized as follows: Section 3.2 presents the related work on load balancing and also shortly introduces the underlying MAC protocol, LMAC, which gives us the possibility to use the cross-layer information. Section 3.3 gives the details of Partition-based Network Load Balanced routing protocol (P-NLB) and its two-level approach. In Section 3.4, P-NLB is compared with two existing routing protocols, i.e. a centralized protocol, and the basic shortest path routing (SPR) by extensive simulations. The networking performance of different routing metric are also compared in Section 3.4. Finally, Section 3.5 draws the conclusions.

3.2 Related Work

In this section, we first survey briefly the existing works on load balancing in multi-sink WSNs. We then introduce the LMAC Protocol which is used in our load balancing protocol as the underlying MAC protocol that provides cross-layer information.

3.2.1 Multi-sink Routing and Load Balancing

As the size of the network increases, the sensors near the sink nodes will dissipate energy faster than other sensor nodes far from the sink as the nodes neighboring of a sink need to relay a larger number of messages. Therefore, these nodes become critical points of the network since the sink will be isolated from the network when they run out of their energy. One promising approach is to deploy multiple sink nodes in WSN, since it can more evenly distribute the load, thus the energy consumption of sensors and also improve the scalability of the networks. The benefits of having multiple sinks in the network are also discussed in Chapter 2.1. In a multi-sink deployment, there are two main issues needed to be addressed: (i) Load Balancing between sinks in the network, and (ii) Load Balancing between sensors in each partition.

To achieve efficient routing in multi-sink WSNs, a common routing technique is building multiple spanning trees in which sensors are vertices and forwarding vectors are edges. Each spanning tree has one sink which is the root of that tree. Such a tree is also called *cluster* or *partition* in the literature. Load Balancing between sinks is generally equivalent to the problem of efficient clustering where each cluster has a data sink. Various techniques have been proposed in the literature for network-wide load balancing. In *Load Balanced Clustering* (LBC) [73] and *Greedy Load-Balanced Clustering Algorithm* (GLBCA) [112], the distribution of the load is controlled by creating clusters in the network. Each cluster in the network has a cluster head which gathers data from sensors within the cluster. In LBC and GLBCA, the network contains multiple sinks, each of which is also a cluster head. LBC uses energy reserves and locations of sensors to balance load among sinks. In GLBCA, the authors define the problem of balancing the load among sinks as an optimization problem with the objective of minimizing the maximum load of each sink in a given network and prove that under general conditions this is an NP-hard problem.

There are several reasons for balancing the load also over the sensor nodes more uniformly in each cluster, i.e. reducing congestion in nodes, and extending the lifetime of the network nodes. Spanning trees for routing are used by most of the existing works [175, 42, 51]. Each sensor other than sink has a pointer to its parent which is one of its neighbors. The procedure of deciding which of the neighboring sensor nodes will be the current parent of a sensor node is called *parent selection*. Parent selection problem [175] is directly related with load balancing between sensors in each cluster. In *Distributed* algorithm for *Load Balanced Clustering* (DLBR) [42], the goal of distributing energy consumption is achieved by looking at the energy level of neighbors and forwarding packets to the node (i.e. parent node) which has a high energy level, while avoiding to forward packets to the nodes which are nearly depleted. In *Node-Centric Load Balancing* (NCLB) [51], the authors look at the structure of the routing paths from sensor nodes to a single sink and use an offline method for balancing the load across different branches of the routing tree rooted at this sink. In this offline algorithm, the routing tree is built step by step. At the start of the algorithm, only the sink and its one-hop neighbors are part of the spanning tree. At each iteration the “weight” (load) of the branches and the “freedom” (a measure of how much space the branch has for expansion) of the branches are calculated and lightest branch with the most freedom is expanded. They use the “Chebyshev Sum Inequality” as a load balancing metric. Both DLBR and NCLB have only one sink in the network and try to balance the load between sensors in the routing tree rooted at that sink. The *Energy Efficient Distributed Dynamic Diffusion*

Table 3.1: *Overview of the related works on load balancing in WSNs*

Protocol	**Type**	**Objective**	**Localization usage**
LBC	Clustering	Centralized Load Balancing between sinks	✓
GLBCA	Clustering	Centralized Load Balancing between sinks	✓
NCLB	Routing	Centralized Load Balancing in a cluster	–
DLBR	Routing	Distributed Load Balancing in a cluster	–
e3D	Routing	Distributed Load Balancing in a cluster	✓
RTLD	Routing	Distributed Load Balancing in a cluster	✓

(e3D) [137] protocol uses the distances between each node and the distances between each node and the sink as a metric for forwarding data from a node to the sink, directly or via multiple other nodes. In this diffusion based approach a node can order – via special control packets – other nodes to stop using it as a relay node if, for instance, the message queue is full or the energy level is below a certain threshold. *Real-Time Routing Protocol with Load Distribution* (RTLD) [26] uses geodirectional-cast forwarding for real-time communication in WSN. Its routing depends on optimal forwarding decisions that take into account of the link quality, packet delay time and the remaining power of next hop neighbors.

Another method to distribute the load more evenly among the sensor nodes in each cluster is using multi-path routing. Several multi-path routing schemes [49, 156] have been proposed in the literature. Generally, multi-path routing approaches tend to be an extension of the single shortest-path routing paradigm with use of additional paths. In multi-path routing schemes designed for load balancing, sensors, which are on the original primary path between source and sink nodes, switch their parent for different transmissions to forward data from different paths. Switching parent nodes can be done by periodically adaptation of the routing trees in the network. This adaptation is based on some quality metrics of each neighbor, such as distance from sink, and number of paths passing through the node [49].

Table 3.1 gives an overview on the related works discussed above. The existing works show some useful ideas for load balanced routing strategies in WSNs. Multi-sink routing can be done using multi-path to a multiple sinks, such as in [49, 156], in order to increase the chance of successful packet delivery. As a drawback, this strategy increases the delivery costs, since in fact multiple packets are sent while only one packet has to reach its destination. Depending on the network situation the extra cost may be higher than the increase in performance. The protocols in [73, 51, 112] are centralized which makes them not scalable for large networks. Also, they are not flexible in topology and network condition changes. Load balancing between sensors shows promising results in single sink networks and it is expected that it performs even better in multi-sink networks, where the load can be distributed over more sinks. On the other hand, none of the related works provides a distributed solution which combines global load balancing among sinks with metric-based convergecasting for load balancing within the partitions. As shown in the table, some of the techniques use location information of the sinks and the sensors for clustering and load balancing in each cluster. In our approach, we do not use location information of the nodes. In addition, different than the existing works, we use cross-layer information for reducing communication cost to achieve load balancing in multi-sink WSNs.

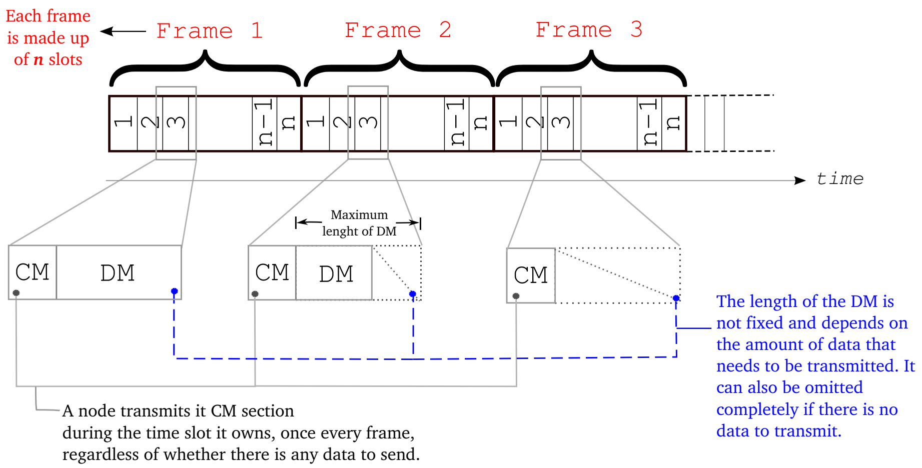

Each frame is made up of *n* slots
Frame 1
Frame 2
Frame 3
1
2
3
n−1
n
time
Maximum lenght of DM
CM
DM
The length of the DM is not fixed and depends on the amount of data that needs to be transmitted. It can also be omitted completely if there is no data to transmit.
A node transmits it CM section during the time slot it owns, once every frame, regardless of whether there is any data to send.

Figure 3.1: *Illustration of frames and slots in LMAC [40]*

3.2.2 LMAC Protocol

The Data link layer of WSN protocol stack contains the Medium Access Control (MAC) and Logical Link Control (LLC) functions. The MAC protocol is important for the work presented in this chapter, since cross-layer information of the MAC is used in the proposed routing protocol. The Light-weighted Medium Access Control (LMAC) [163] protocol is used as underlying MAC protocol in this chapter. Since these layers are close to each other in WSNs, LMAC in the data link layer is able to pass useful information to the network layer. Therefore, the MAC protocol used in this chapter – LMAC – is also described. Although LMAC is used as the underlying MAC protocol, every other MAC protocol which provides the same information to the network layer can be used.

LMAC is a scheduled based protocol designed for WSNs using a combination of Time Division Multiple Access (TDMA) and Space Division Multiple Access (SDMA) techniques. It functions without a central manager; nodes function autonomously. As shown in Figure 3.1, time in LMAC is divided into frames, each of which is further divided into a fixed number of time slots. Every node chooses its own slot using a distributed algorithm that uses only locally available information. Every node is allowed to pick any slot per frame to transmit data to other nodes as long as the chosen slot is not owned by any other node within two-hop neighborhood. The major advantage of such a TDMA scheme above contention based schemes is the lack of collisions. Only one node will transmit during a time slot in a frame, so no collisions occur, which are a source of energy waste. This mechanism also efficiently helps to avoid the hidden-terminal problem as it makes it impossible for two nodes, which are two hops away from each other, to transmit at the same time.

A time slot consists of two parts, a Control Message (CM) section and a Data Message (DM) section. The CM contains control information about the node and its one-hop neighborhood, as shown in Table 3.2, and has a fixed length. It is broadcast by a node to its neighbors during its own time slot once every frame, irrespective of whether the node has any data to send. A new node joining the network first listens out for the CMs of all its

Table 3.2: *Contents of the Control Message (CM)*

Description	Size (bits)
Node identification	16
Current slot	5
Distance to sink	8
Occupied slots	32
Collision in slot	5
Parent ID	16
Destination Sink	16
Routing path length	32

Table 3.3: *Additional fields in CM for metric-based routing*

Routing Metrics	Size (bits)
Number of child nodes	8
Number of descendant nodes	8
Buffer occupancy	7
Energy level	5

neighbors and then picks one of the slots that is marked as unoccupied. The DM section is used for sending data. In the CM section a node has addressed the receiver node, and in the DM section it transmits the data. The addressed node determined by the CM section of the transmitting node knows that the data is for it and listens for the data in the DM section. Other nodes that are not addressed by the transmitting node can switch off their transmitters, thereby save energy.

In this chapter, we utilize the CM section for metric-based routing in WSNs. The extra information shown in Table 3.3 can be added to CM depending on the routing metric defined by the application.

3.3 Partition-based Network Load Balanced Routing

In this section, we present a cross-layer approach to solve the load balancing problem in a basic scenario, i.e. a static multi-sink wireless sensor network. Our approach is called *Partition-based Network Load Balanced* routing (P-NLB), and is designed for achieving load balancing in large scale static multi-sink WSNs. We first describe the terms and assumptions used in the protocol design. We then explain the details of P-NLB.

3.3.1 Terms and Assumptions

The following terms are important for understanding the operations of a tree-based routing protocol in a wireless sensor network. Some of them are illustrated in Figure 3.2. The network is a representation of a graph G, with vertices V as nodes (sensor nodes and data sinks) and edges E as communication links. N is the number sensor nodes, M the number of data sinks.

- Sensor nodes – Device with low processing and memory capacity, and limited power supply.
- Data sink – It has higher capacity than common sensor nodes: more processing power, unlimited power supply. It is connected to the end-user application.
- Communication link – It is a bidirectional link between two sensor nodes that is used for exchanging information. There is a communication link between a pair of nodes if

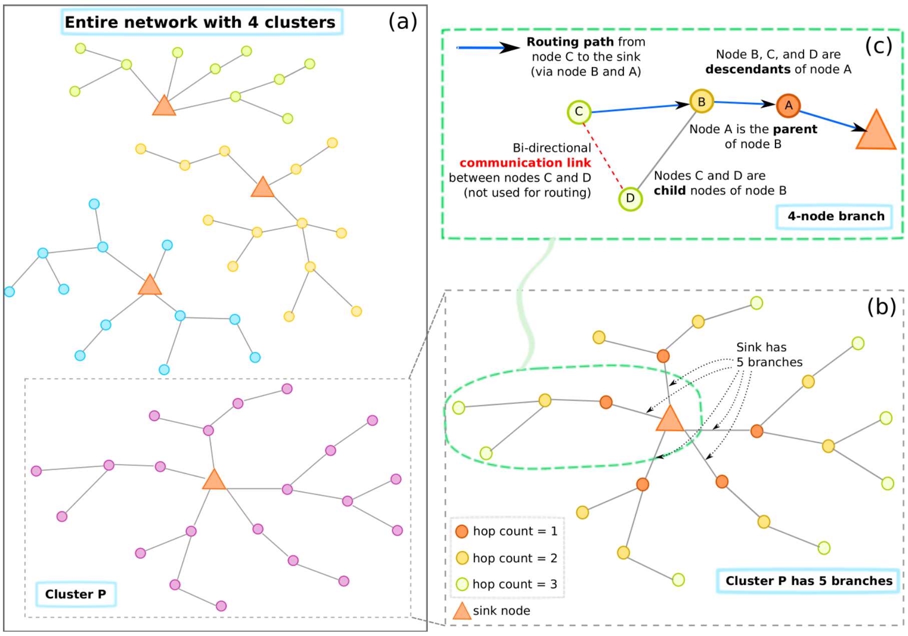

Entire network with 4 clusters
(a)
Cluster P
(c)
Routing path from node C to the sink (via node B and A)
Node B, C, and D are descendants of node A
C
B
A
D
Node A is the parent of node B
Bi-directional communication link between nodes C and D (not used for routing)
Nodes C and D are child nodes of node B
4-node branch
(b)
Sink has 5 branches
hop count = 1
hop count = 2
hop count = 3
sink node
Cluster P has 5 branches

Figure 3.2: *WSN components in a tree-based routing scenario*

they are within transmission range of each other. The term is sometimes abbreviated as link.

- Neighbor – Two nodes are neighbors of each other if there is a communication link between the two nodes.
- Hop count – It is the shortest distance between a node and a sink, measured in hops (see Figure 3.2(b)). A packet travels two hops if it travels to a node via another node.
- Child and Parent nodes – Each sensor node has a vector pointing to a neighbor node, representing to which neighbor a data packet is forwarded. The sending node is the child node, the receiving node is the parent node (see Figure 3.2(c)).
- Spanning tree – A spanning tree of a connected, undirected graph G is a selection of edges (i.e. links) of G that form a tree spanning every vertex (i.e. sensor node). In case of multiple sinks, multiple spanning trees are formed. All nodes of a spanning tree form a *cluster* (see Figure 3.2(a)).
- Routing path – It is a path which packets use to travel from source node to the data sink.
- Descendants – The descendant nodes (sometime called downstream nodes) of a node are the nodes that are on the same routing path, but have a higher hop count – in other words are further away – from that specific node (see Figure 3.2(c)).

- Branch – A sink has one or more neighbors; these neighbors are called top-level nodes. These nodes are the beginning (i.e. the root) of a top-level branch (sometimes called just branch).

The work presented in this chapter is targeted for applications such as environmental monitoring where a large number of data is sent periodically towards multiple static sinks in the network. For example, the application mentioned in Section 2.3 describes a sensor network laid out to monitor various physical parameters in a forest. Based on this application, we have made the following assumptions about the network:

- Sinks and sensor nodes in the network are stationary; they do not change their position.
- Communication links between pairs of nodes are bi-directional and these links between nodes do not change over time. The radius of the transmission range of nodes is much smaller than the size of the area where nodes are deployed, therefore direct communication between all nodes in the network is not possible.
- There is only one pattern of data flow in the network: from sensor nodes to data sinks. This is the most common communication paradigm in (data gathering) sensor networks. Packets (i.e. queries) sent from data sinks to specific sensor nodes in the network is not under consideration of this work.
- Sinks can (directly) communicate with each other using a high-speed communication channel.
- Sinks are equal from the information point of view; it does not matter to which sink a data packet is sent. We assume that after reception of the packets all sinks forward them to the same end-user application.
- It is possible to exchange cross-layer information between data-link and network layers, such as information about neighboring nodes.
- No data aggregation is done by nodes in the network.

3.3.2 A Two-Level Approach

P-NLB protocol exploits a two-level approach that combines metric-based routing on *local level* with a network wide load balancing technique among sinks on the *global level*. On the local level, sensors exchange information with their one-hop neighbors and get a view of their local neighborhood. This local neighborhood information is provided via cross-layer communication with the MAC-layer which is implemented by LMAC protocol (Section 3.2.2) in this work.

On the *global level*, a clustering technique is used in order to spread the load in the network uniformly among all sinks. The novel parts of this mechanism is that no explicit clustering phase is used, but the nodes in the network achieve clustering on the global level by an intelligent routing on the local level. On the global level, the sinks determine the structure and cluster sizes of the network (provided by the LMAC protocol) and inform the sensor nodes about the network structure. To be more specific, the network structure is described by the information if the clusters in the network are balanced or not, which cluster

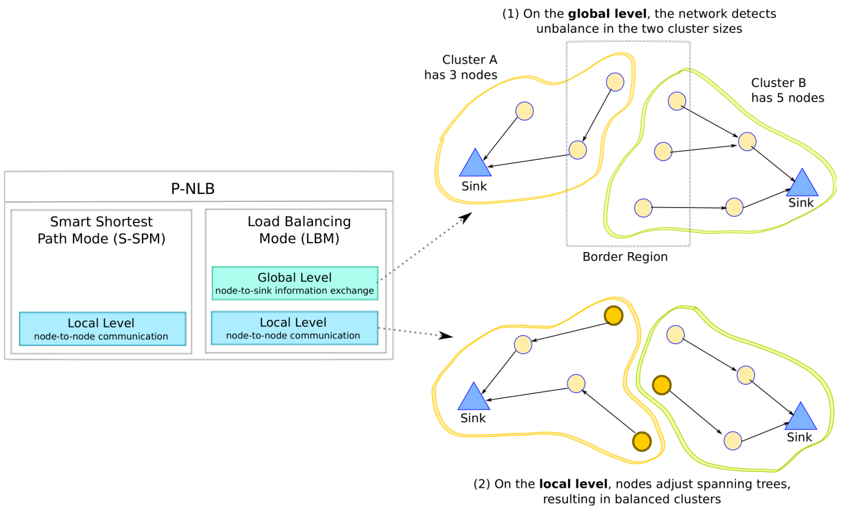

(1) On the **global level**, the network detects unbalance in the two cluster sizes
Cluster A has 3 nodes
Cluster B has 5 nodes
Sink
Sink
Border Region
P-NLB
Smart Shortest Path Mode (S-SPM)
Local Level node-to-node communication
Load Balancing Mode (LBM)
Global Level node-to-sink information exchange
Local Level node-to-node communication
Sink
Sink
(2) On the **local level**, nodes adjust spanning trees, resulting in balanced clusters

Figure 3.3: *Two-Level Routing Approach*

is the smallest. On the local level, the sensor nodes use the information provided by the sinks in combination with the local information to make their routing decisions. Thus, the global level does not do the actual clustering or routing. It just gathers information from the network and informs the nodes about this information.

The *local level* does the actual routing and clustering and uses a metric-based routing mechanism where every node decides what the next hop is for creating a routing path towards a sink. This decision is done by selecting a neighbor as the parent node, which forms an edge in the spanning tree. The resulting spanning tree is used for routing the packets from nodes to sinks. The decision of selecting a neighbor as the parent node depends on the information provided by the sinks on the global level – in case of the global balancing mode – and the routing strategy defined by corresponding routing metric of the network. The routing strategy of a network, such as *avoiding congestion*, depends on the demands of the application running on the WSN. On the local level nodes use the cross-layered approach to exchange information with their one-hop neighbors and get a view of their local neighborhood (provided by the LMAC protocol). An illustration of this two level routing approach is given in Figure 3.3.

3.3.3 Protocol Phases

P-NLB protocol consists of a *setup* phase and an *operational* phase, which are explained in the remainder of this section.

Setup Phase

The initial clustering is done by the simplest approach in which each sensor node in the network is connected to the nearest sink (in terms of hop count) at the network initialization. To achieve this every sink floods a message through the network. As result, each sensor node knows its hop count to each sink in the network. All nodes then send a message to the nearest

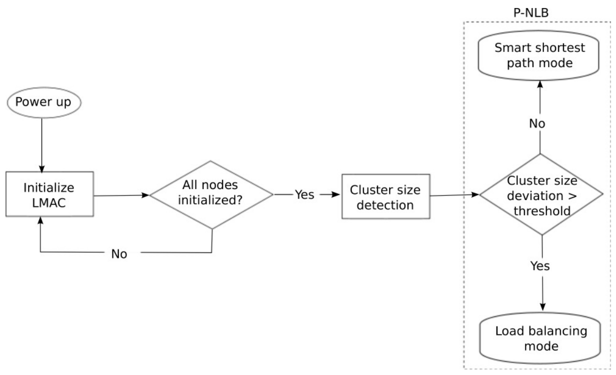


Figure 3.4: *State diagram of the setup phase*

sink for informing the sink about their presence. In case of multiple nearest sinks, the node randomly chooses only one of the sinks. Thus, every node belongs to only one cluster.

The state diagram of the setup phase is illustrated in Figure 3.4. In the setup phase, all the nodes in the network initialize themselves with LMAC protocol. They register to the network, learn about the neighbors in their proximity, get their hop count the sink and finally acquire an LMAC time slot. This phase is important for the nodes, because they acquire valuable information about their local network neighborhood, which will be used in the operational phase in order to start efficient routing. After the setup phase has ended, the sinks in the network have information about the initial cluster sizes in the network, which is useful for determining the actual need of balancing the network. In the cluster size detection state, all sinks exchange information about the sizes of their clusters. After receiving the cluster size information from the other sensors, a sink calculates the cluster size dispersion in the network. For instance, a high standard deviation indicates that the nodes in the network are not equally distributed between the sinks. After the cluster size detection state, the network enters the operational phase in which the cluster size distribution information will be used to determine the appropriate actions.

Operational Phase

In the operational phase, dynamic spanning trees for each sink are established based on the neighborhood information of sensor nodes acquired in the setup phase. These spanning trees are used for routing of data packets from sensors to the sinks. The spanning trees to the sinks are constantly maintained and adjusted to construct the most efficient routing paths in the network. This is done by the nodes as a result of constant updating of their parent nodes.

The state diagram of the operational phase is given in Figure 3.5. The steps belonging to the global level are colored blue, while the steps belonging to the local level are colored yellow. The state diagram illustrates that the smart shortest path mode (S-SPM) has only local level, while the load balancing mode (LBM) has also global level. The global and local levels of the protocol are further explained in the next sections.

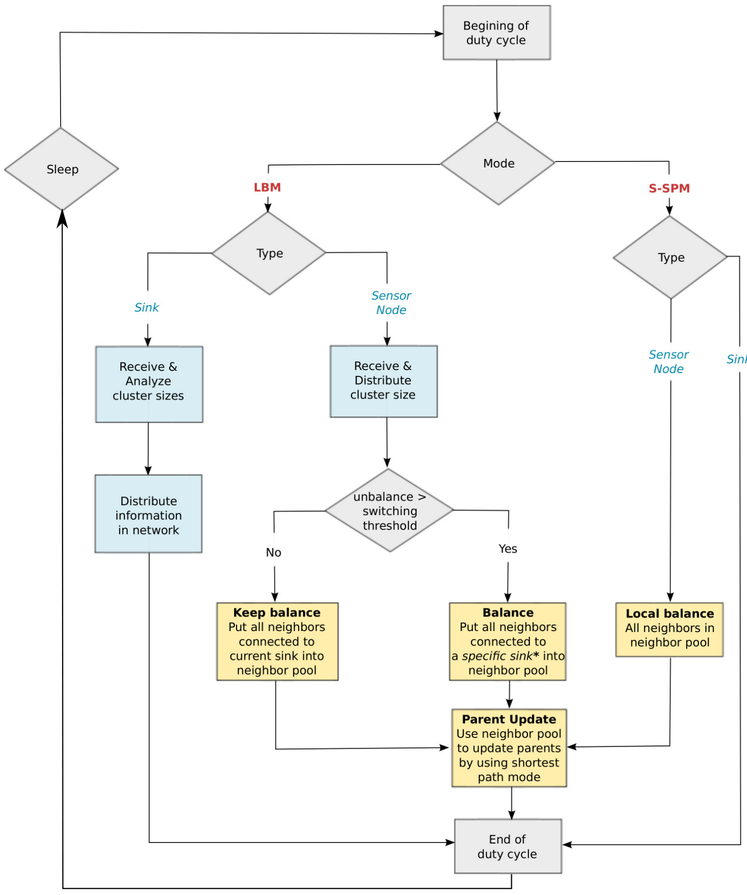

Begining of duty cycle
Mode
LBM
S-SPM
Sleep
Type
Sink
Sensor Node
Receive & Analyze cluster sizes
Receive & Distribute cluster size
Distribute information in network
unbalance > switching threshold
No
Yes
Keep balance Put all neighbors connected to current sink into neighbor pool
Balance Put all neighbors connected to a specific sink* into neighbor pool
Local balance All neighbors in neighbor pool
Parent Update Use neighbor pool to update parents by using shortest path mode
End of duty cycle

Figure 3.5: *State diagram of the operational phase (*The selection of the sink (cluster) is explained in Section 3.3.5)*

3.3.4 Global Level – Cluster Information Gathering and Distribution

The aim of the global level is gathering and distributing useful information from the network to achieve establishment of multiple non-overlapping clusters (i.e. spanning trees) in the local level where each spanning tree has a sink as the root node of the tree. If all the spanning trees in the network contain more or less the same number of nodes, then the network load is balanced, assuming the traffic load generated by each sensor node is more or less equal.

In order to keep the network load uniformly distributed over the sinks, the sinks need to know what the actual network load is. Network load of a cluster is determined by the number of nodes in this cluster, or in other words, cluster size. The sinks give the information of their cluster sizes to the nodes in the network. The mechanism of collecting the cluster size information and distribution of it to the nodes in the network has three steps.

- *Information gathering* – Information gathering is done in the *setup phase* which is explained in Section 3.3.3. Nodes keep track of the number of child nodes they have. Each node propagates the number of child nodes to its parent in the route to the sink which is at the root of the node's current cluster. In this way, each sink knows what the number of nodes in its spanning tree is and thus the cluster size.
- *Analyzing* – Assuming (direct) communication between sinks, each sink gets the information about all other cluster sizes in the network and consequently the current load distribution in the network.
- *Distribution* – Each sink distributes the cluster size information (i.e. the sink id and its cluster size) back into the network by using cross-layer communication via LMAC protocol.

In some cases, the difference between cluster sizes can be very small and unnecessary load balancing operations should be avoided. This is needed to avoid *oscillation* of nodes – close to the border regions of their clusters – between clusters which have very similar cluster sizes. In small networks, this oscillation has not much effect, but in larger networks which have more nodes in the middle of two clusters, oscillation of nodes may cause instability and result in decreased performance. To avoid the oscillation effect, a parameter called *switching threshold* (STh) is introduced. The switching threshold helps to stop nodes from attempting to balance slightly unbalanced networks. As illustrated in Figure 3.5, if the differences between cluster sizes is bigger than switching threshold, then the network starts to operate in Load Balancing Mode. The effect of switching threshold on the routing tree construction is explained in Section 3.3.5.

The global level procedure described above is a continuous process of gathering, analyzing and distributing the information. The drawback of this continuous process is that the global information may not be always very up-to-date since the information distribution takes time to reach all the nodes in the network. As a result of non-up-to-date information, nodes might take not the best decisions and sub-optimal spanning trees are built in the network. On the other hand, P-NLB uses a distributed approach where nodes themselves decide to which sink they route via which parent node. Construction of the clusters is not controlled by the sinks. Sinks in the network only have a task of sending periodically information about the cluster sizes into the network for enabling load balancing between sinks. This is done by using the CM section of LMAC protocol and requires no broadcasting, keeping this mechanism scalable.

3.3.5 Local level – Metric-based Tree Building

A common routing approach in WSNs is the *Shortest Path Routing* (SPR) [62] paradigm to send data packets to the sinks. SPR is defined as the routing mechanism where a node forwards its data only to the neighbor which has the shortest distance – measured in hop counts – to the sink. This results in a loop-free Minimum Spanning Tree (MST) rooted at a sink [98]. It minimizes the number of hops a packet travels, leading to the formation of spanning trees containing different amount of sensors, since selecting the shortest path does not account for the effect of load aggregation on upstream links. Therefore, by assuming uniformly generated load per node, SPR creates spanning trees with different loads in the network. Instead of the randomized packet forwarding to any neighbor closer towards a sink, we use well-defined routing metrics to increase the efficiency of SPR, similar to the approach in [175]. The idea is basically choosing a parent node based on other metrics when a node has several same length paths. In addition, this decision should take the existence of multiple sinks into account to achieve both inter-cluster and intra-cluster load balancing.

In the local level, every node decides itself what the next hop is to create a routing path to a sink. The selected neighbor is called the *parent node* and the node itself is the *child node*. All these small one-hop connections will result in one long routing path from source node to the sink. All the routing paths from all source nodes to a specific sink form the spanning tree, also called routing tree, for that sink. The decision of selecting a neighbor as the parent node depends on the information provided by the sinks on the global level and the corresponding routing metric of the network. On the local level, sensors use one-hop neighborhood information provided by LMAC to analyze their neighbors and select the best neighbor as its new parent based on the routing metric. The additional fields of a control message, which are shown in Table 3.3, are used for metric-based routing. These metrics are (i) Number of child nodes, (ii) Number of descendant nodes, (iii) Buffer occupancy, and (iv) Energy level of the node.

Different applications have different demands from the network, such as long lifetime, low message latency or high throughput. Based on these demands different routing strategies might be used. A routing strategy can be avoiding low-energy nodes, avoiding congested nodes, routing to the closest sink. Different routing strategies might be used separately in a single network, if there are high- and low-priority messages for example. Based on the routing strategy, a node has to choose a routing metric to select the best parent and to form a spanning tree in the network.

Routing Metrics

Four routing metrics are used for comparison in this work, although it is possible to define more, e.g. link quality, link usage, neighbor distance. In the performance evaluations, simulations clarify which routing metric fits best for which demand of the application.

- *Child nodes* – A node receives packets from each child node within one frame in LMAC. Having many child nodes has some negative consequences for this node. Firstly, every time a node receives data from its child nodes, its transceiver consumes energy and when it forwards the data, its transceiver is again activated. If a node has many child nodes, it is likely that it consumes more energy and its energy source gets depleted earlier than the nodes having less child nodes. Secondly, the data coming from child nodes sometimes needs to be stored in the buffer before it is forwarded to the parent node. Receiving more data than the node is able to transmit results in full buffers and eventually buffer overflows. Buffer overflow may result in packet drops. *Selecting the neighbor having the smallest number of child nodes as the parent node* can help to solve these problems.

- *Descendant nodes* – This metric is related to the previously mentioned metric, child nodes, but differs slightly. Whereas the metric of number of child nodes takes into account only the one-hop neighbors, descendant nodes are all nodes down on that branch of the routing tree (see Figure 3.2(c)). This mainly affects the total amount of traffic a node has to process. A node with many descendant nodes, all generating data packets, has to receive and forward all these data up in the tree. *Selecting the neighbor having the smallest number of descendant nodes as the parent node* reduces the load of heavily loaded parent nodes.

- *Remaining Energy level* – After a certain uptime of the network, some nodes might have to forward/transmit a lot of data; this will drain their energy source. Other nodes might stay in an idle state for a long time, or transmit only hardly any data. It is wise to shift data traffic from nodes having near empty energy buffers to nodes with full energy buffers in order to extend the total network lifetime. *Selecting neighbor with highest remaining energy level as the parent node* results in avoiding nearly depleted nodes in the routing path.

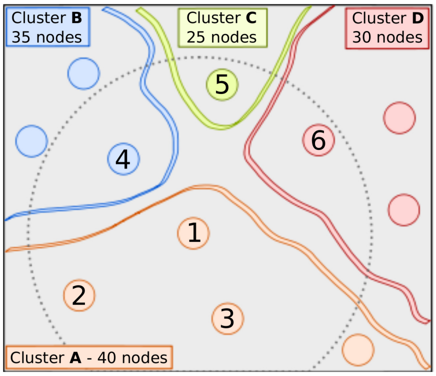
- *Buffer* – Every node has a message buffer where it stores the incoming packets, before they are forwarded to the parent node. Assuming all packets have the same priority, and a first in first out strategy is used, the latency of packets increases when there are many packets in the message buffer of a node. *Select neighbor with the least amount of packets in the packet queue as the parent node* can help to reduce congestion and latency in the network.

Parent Selection

Since we assume the network is a connected graph, all nodes have at least one neighbor. However, in most cases every node has several neighbors, which all can be selected as the parent node for forwarding data. In this section, we introduce a metric-based parent selection mechanism based on shortest path routing (SPR). Parent selection in SPR is enhanced by adding an extra step, *neighbor pool construction*, to the beginning of the parent selection process. The enhanced SPR is called *Smart Shortest Path Mode* (S-SPM). Smart here only means that if a node has more than one neighboring sensor node as parent candidate, S-SPM makes the parent selection decision based on some application-specific routing metrics such as energy level, buffer capacity, congestion avoidance, etc.

A neighbor pool consists of parent candidates of a specific node. To define the neighbor pool, a node uses the local information (e.g. hop counts of neighbors) and the global information (i.e. cluster sizes of sinks). A node can choose to leave its current routing tree (cluster) and join another cluster by selecting a neighbor as its new parent, which is in the other cluster. Nodes having neighbors from other clusters are generally located near the cluster borders (see Figure 3.3). By joining a smaller neighboring cluster, a node decreases the size of its own cluster and increases the size of its new cluster. Thus, it makes one step forward to balance the load between clusters. The neighbor pool construction mechanism filters some neighbors of a node according to the cluster sizes before the metric-based parent selection takes place.

The filtering of neighbors and neighbor pool construction based on the cluster sizes is illustrated in Figure 3.6. In the figure, Node 1 in Cluster A is searching for parent candidates. If the network is only in S-SPM, it does not care about global load balancing among sinks; therefore, a node searching for an appropriate parent adds all neighbors to its neighbor pool (see $S-SPM$ column in the Figure 3.6). If the network is in LBM, neighbor pool construction is done by considering the switching threshold. If the difference between the cluster size of



Neighbor Table and Neighbor Pool of Node 1

Node	Cluster	Cluster Size	S-SPM	LBM	
			Add to Pool	Add to Pool	
			(No global balancing)	STh < Diff	STh > Diff (No global balancing)
2	A	40	✓	✗	✓
3	A	40	✓	✗	✓
4	B	35	✓	✗	✗
5	C	25	✓	✓	✗
6	D	30	✓	✗	✗
				STh=10	STh=20

STh: Switching Threshold
Diff: The difference between cluster size of Cluster A and the cluster size of the neighboring cluster

Figure 3.6: *Neighbor Pool Construction for Parent Selection*

a node and the cluster size of any neighboring cluster of this node is smaller than switching threshold, there will be no attempt to balance load between these clusters. The node will add neighbors to the pool if they are in the same cluster with the node (see LBM, $STh > Diff$ column with $STh = 20$ in Figure 3.6). If the difference between two cluster sizes are bigger than the switching threshold, then the load balancing is needed and neighbor nodes in the neighboring cluster having the smallest size will be added to the pool (see LBM, $STh < Diff$ column with $STh = 10$ in Figure 3.6).

After the construction of neighbor pool, one of the neighbor nodes from the pool is chosen as the parent node. This choice is done based on the given routing metric. The following steps are executed for selecting a parent:

- *Step 1:* Node checks the hop count of the nodes in the pool and considers only the neighbors which have the smallest hop count as parent candidate in the next step.
- *Step 2:* Node applies the given routing metric on the remaining neighbor nodes to find the parent node. For instance, if the routing metric is *buffer*, the node only considers the neighbor nodes that have the least number of packets in their buffers.
- *Step 3:* If all remaining neighbors have the same properties, node chooses randomly one of them as the parent node.

Shortest Path Relaxation

The shortest path paradigm can be relaxed to have a more flexible routing tree construction. Instead of always selecting a neighbor that is closer to a sink in the network (i.e. Step 1), a node might select a neighbor which has the same hop count as the node itself has. With this small relaxation of the shortest path constraint, bottlenecks can be better avoided in the network. The cost of this relaxation is also small, a slightly longer routing path, and the need for some small precautions in order to avoid loops[*] in the network. An example of this shortest path routing relaxation is illustrated in Figure 3.7. The effect of the relaxation

[*]A simple method for loop avoidance is presented in [131].

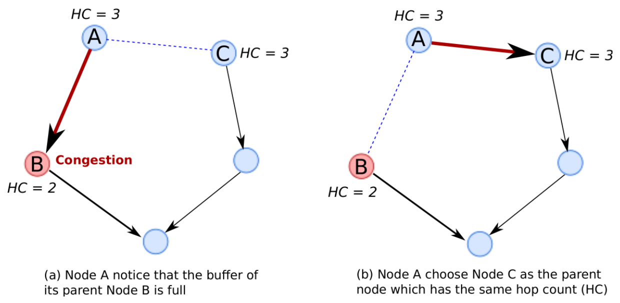

HC = 3
A
C
HC = 3
B
Congestion
HC = 2
(a) Node A notice that the buffer of its parent Node B is full
HC = 3
A
C
HC = 3
B
HC = 2
(b) Node A choose Node C as the parent node which has the same hop count (HC)

Figure 3.7: *Shortest Path Relaxation*

of shortest path on the performance of the protocol is analyzed by extensive simulations in Section 3.4.4.

3.4 Performance Evaluation

Simulations are performed using MATLAB as programming tool. The basic SPR, NCLB, and two modes of P-NLB are implemented and compared in MATLAB simulations. LMAC protocol is used as the underlying MAC for all protocols. 200 random connected networks consisting of 64 static sensor nodes and 2 static sink nodes are established for the *routing metrics performance simulations* and 1 to 5 sinks are deployed for the *multi-sink performance simulations*. The simulations are run for 5000 MAC frames, so each sensor has 5000 opportunities of performing actions of generating and sending data. Every sensor generates 1 packet every 6 frames. All nodes in the network generate the same amount of traffic. This is a common situation in WSN designed for environmental monitoring, where data packets only consist of fixed size sensor readings. Number of time slots per frame and the degree of the network are closely related to each other. The number of time slots per frame is set to 16. Nodes have a packet buffer able to contain 8 packets. These simulation results show the performance of load balancing and routing algorithms.

3.4.1 Evaluation Metrics

In order to evaluate the performance of P-NLB, we define a set of performance evaluation metrics:

- *Average packet delivery latency* – The *end-to-end delay* between sending a packet from the source node and receiving the packet at a data sink.

- *Network lifetime* – The time from initialization of the network till the first node fails due to energy depletion.

- *Throughput* – The amount of data a network processes (per unit of time, frame length, for example). It is measured as the number of packets arriving at the sinks during one frame. Best effort routing is used which means no resending of lost packets. We measure the average throughput of the network represented by the following equation

$$Throughput = \frac{\sum_{i=1}^{\tau}(\sum_{j=1}^{\eta} R_j^i)}{\tau} \tag{3.1}$$

where τ is the simulation duration in frames, η is the number of data sinks in the network, and R_j^i is the number of packets received by sink j in time frame i.

- *Packet Delivery Ratio (PDR)* – The ratio of the number of packets delivered at the sinks to the total number of packets generated by the sensor nodes.
- *Load balance* – Standard deviation of the load on sinks in the network. The standard deviation of the sink load, σ, is given in the following formula

$$\sigma = \sqrt{\frac{1}{\eta} \times \sum_{j=1}^{\eta}(l_j - \bar{l})^2} \tag{3.2}$$

where η is the number of sinks, l_j is the load of sink j, and $\bar{l} = (\frac{1}{\eta} \times \sum_{j=1}^{\eta} l_j)$ is the average load of all sinks.

Both routing modes of P-NLB, i.e. S-SPM and LBM, are simulated. In the S-SPM routing mode simulations, the network always route packets using S-SPM without load balancing. In the LBM routing mode, nodes use the cluster size distribution detection mechanism in the setup phase to enter the operational phase in S-SPM without load balancing or LBM with load balancing. Besides those two routing modes of P-NLB, basic SPR and the Node Centric Load Balancing (NCLB) [51] protocol are also implemented in our simulator and compared with P-NLB in the simulation results. SPR acts as a lower bound reference of what the performance of a basic not optimized routing algorithm would be. NCLB is a centralized algorithm and adding it to the simulations allows a comparison of a distributed algorithms with a centralized one. However, as mentioned before, the load balancing problem is an NP-hard problem, and NCLB provides no hard upper bound, but only an approximation.

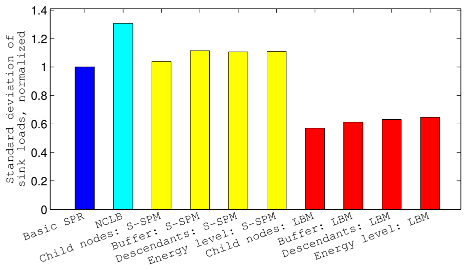
First, we analyze the performance of different routing metrics (i.e. energy level, child nodes, descendant nodes, buffer) discussed in Section 3.3.5 in two different network topologies. Secondly, we test the influence of the number of sinks on the performance of the network. Finally, we evaluate the influence of shortest path relaxation on the networking performance.

3.4.2 Routing Metric Performance

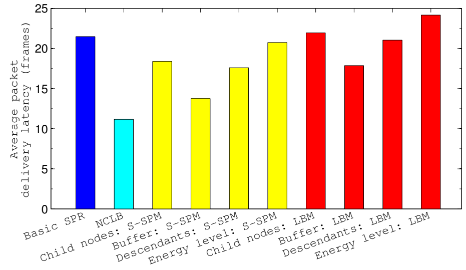
In this set of simulations, we consider two different network topologies: (i) Random topology, and (ii) Asymmetric topology which features a small and a large cluster. In the asymmetric topology, there are two sinks forming initial clusters of 25 and 40 nodes. 200 connected networks for both types of topologies are created and the results are averaged. We compare the performance of different routing metrics combined with P-NLB in these two types of topologies.

Random topologies

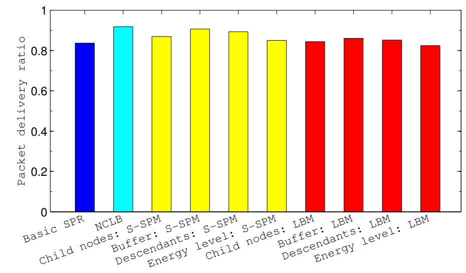
Figure 3.8 shows the performance of all routing metrics combined with local and global load balancing and compares the results also with basic SPR in random topologies. Figure 3.8(a) proves that the global load balancing mechanism, LBM, combined with different routing metrics (i.e. red columns in the figure) can achieve the best load distribution between the sinks in the network. On the other hand, local balancing mode (i.e. S-SPM with yellow columns) alone can not outperform the basic SPR (i.e. dark blue column) in terms of global balancing. It is also clear that all routing metrics achieve more or less the same load balancing for both routing modes of P-NLB. The standard deviation of sink loads achieved by NCLB



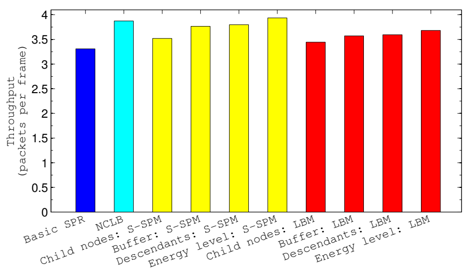
(a) Load Balance



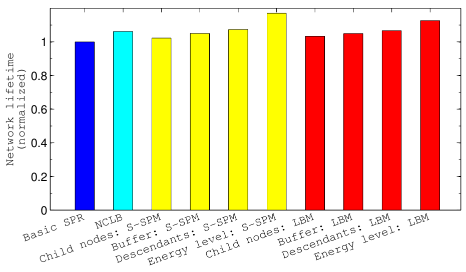
(b) Latency



(c) PDR



(d) Throughput



(e) Network Lifetime

Figure 3.8: *Routing metrics performance in random topologies*

(i.e. light blue column) is the highest since NCLB focuses more on the balancing the load between the branches of the routing trees than the global load balancing between sinks.

Although NCLB is not the best in terms of balancing the load between sinks, it achieves the lowest latency in Figure 3.8(b). The latency of NCLB in random topologies is 50% lower than basic SPR. Latency is in general higher in LBM than in S-SPM. In both routing modes, S-SPM and LBM, the routing metric *buffer* gives the lowest latency among other three routing metrics (i.e. child nodes, descendants, energy level). In S-SPM mode, the latency comes very close to the latency of centralized NCLB.

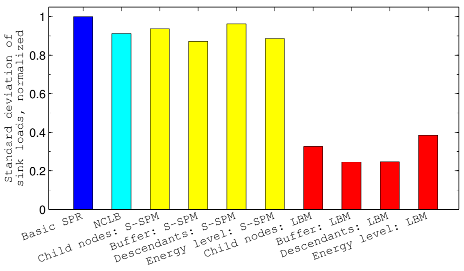
In Figure 3.8(c), NCLB achieves the best packet delivery ratio. However, S-SPM with *buffer occupancy* routing metric is almost the same as PDR of NCLB. Indeed, all results of different strategies are very close to each other. Figure 3.8(d) shows the throughput results. The throughput of basic SPR is lowest of all algorithms whereas NCLB and S-SPM combined with routing metric *energy level* achieves the highest throughput. The throughput of LBM is up to 10% worse than that of S-SPM. As expected routing metric *energy level* combined with S-SPM and LBM achieves the longest network lifetime as shown in Figure 3.8(e). Network lifetime achieved by S-SPM with *energy level* metric is higher up to 10% than SPR and NCLB.

In general latency varies most among the different routing metrics, where routing metric *buffer* performs generally the best in both routing modes of P-NLB. By using this routing metric, nodes are best in avoiding congestion and consequently this results in the lowest latency and highest packet delivery ratio. The results of the other performance metrics show similar results for all routing metrics, although routing metric *energy level* achieves a better throughput and network lifetime. By considering all results, NCLB performs in general the best, with lowest latency and highest PDR. In almost all cases SPR performs worse than all other algorithms.

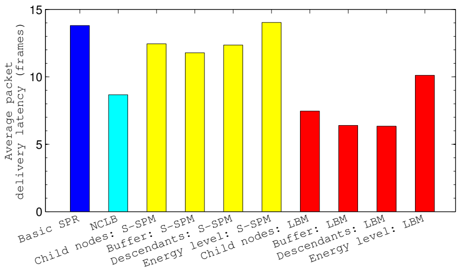
Asymmetric topologies

The networking performances of basic SPR, NCLB, and P-NLB with its two modes, S-SPM and LBM, combined with different routing metrics are shown in Figure 3.9. Global load balancing mechanism LBM is again the best in distributing the load uniformly over the sinks in asymmetric cluster topologies as observed in Figure 3.9(a). In Figure 3.9(b), the effect of better load balancing on the latency can be seen since the LBM combined with *buffer* and *descendants* metrics achieves the lowest latency, it is even lower than latency of NCLB in asymmetric scenarios. The decrease in latency compared to SPR is more than 50%. Routing metric *energy level* has a higher latency in both S-SPM and LBM. This is probably caused by the longer routing paths in order to avoid nearly depleted nodes around the sink.

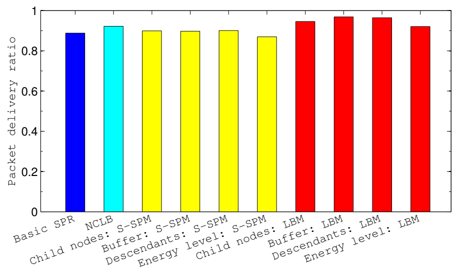
Packet delivery ratio of LBM is slightly (i.e. up to 10%) higher than all other algorithms in Figure 3.9(c). The highest PDR is also achieved by using routing metrics *buffer* and *descendants* in LBM. On the other hand, in Figure 3.9(d) throughput is showing different results than PDR. This can be explained by the fact that traffic load on the top-level neighbors of the sinks is more important for the throughput than up to those nodes. Therefore, it is not a surprise that the routing metrics *buffer*, *descendants* and *energy level*, which can better route packets around the congested top-level neighbors to less congested top-level neighbors, have higher throughputs. As expected, NCLB has the highest throughput, since it makes the best use of the top-level neighbors of the sinks. Figure 3.9(e) shows the network lifetimes



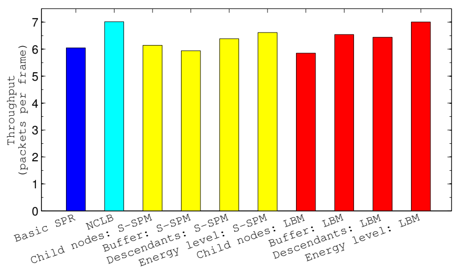
(a) Load Balance



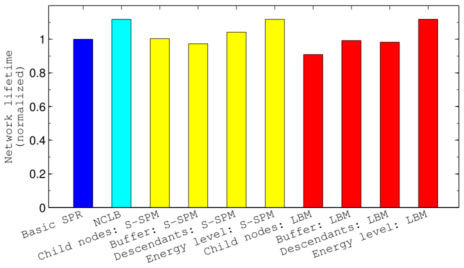
(b) Latency



(c) PDR



(d) Throughput



(e) Network Lifetime

Figure 3.9: *Routing metrics performance with asymmetric clustering topologies*

achieved by all algorithms. As expected, S-SPM and LBM combined with *energy level* metric achieve a network lifetime that is as long as the network lifetime obtained by NCLB. NCLB can also results in a high network lifetime, because it is able to distribute the load over all neighbors of the sinks. Since these neighbors are likely to run out of energy first, this approach extends the lifetime of those nodes. The lifetime of the whole network is increased up to 10% in comparison with SPR.

In general it is obvious that routing metric *child nodes* performs worst of all routing metrics in this asymmetric network type, since most nodes have an equal degree. Therefore, this metric cannot gain any advantage. Routing metric *buffer* is most suitable one if low latency and high delivery ratio requirements are needed by an application. Routing metric *energy level* can be the best for networks requiring a high network lifetime and throughput. In asymmetric network topology, P-NLB is able to outperform the centralized algorithm of NCLB in performance metrics latency and packet delivery ratio and achieve the same performance in network lifetime and throughput.

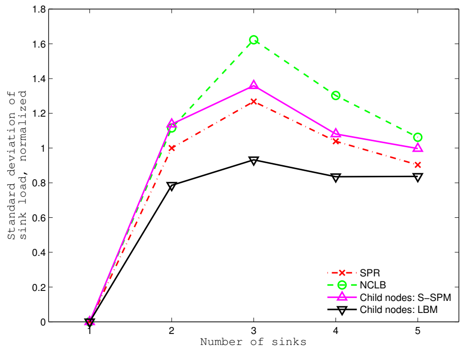
3.4.3 Multi-sink Performance

In this set of simulations, we vary the number of sinks from 1 to 5. P-NLB is combined with the routing metric of number of child nodes. As shown in Figure 3.10 all protocols benefit from an increasing number of sinks in the network. In Figure 3.10(a), it is clearly visible that LBM is much better than the basic SPR and centralized NCLB in terms of uniformly distributing the load over all sinks in the network. However, the difference in load balancing performance between LBM and the other protocols decreases when more sinks are added to the network. The latency results in Figure 3.10(b) show that NCLB starts with a much higher latency than SPR and both modes of P-NLB, but this difference decreases as the number of sinks increases. The latency of LBM is slightly better than basic SPR. When there are 5 sinks in the network, the delivery latencies of all protocols are very close to each other. The reason of this observation is that the average hop count between sinks and sensor nodes decreases when there are more sinks in the network. Although P-NLB balances the load among sinks better than the other approaches, better balancing between sinks does not have much effect on the latency when we have more sinks in the network. This is mainly due to the fact that when there are more sinks in the network, the load difference between sinks gets lower. This observation is also visible in the other graphs. Although LBM balances the load better over the sinks, the throughput and delivery ratio results in Figures 3.10(c) and 3.10(d) do not benefit much from this balancing. The network with one sink is not able to process all data traffic load efficiently in the network; thus, PDR is quite low regardless of the used algorithm. In the networks with multiple sinks, the performance of basic SPR stays behind of other protocols in both delivery ratio and throughput. NCLB achieves a slightly higher data delivery ratio and throughput. Figure 3.10(e) proves that the network lifetime increases when more sinks are added to the network. This is an expected results since the average routing path length is shorter; as a result, less energy is consumed in the network for delivering packets to the sinks.

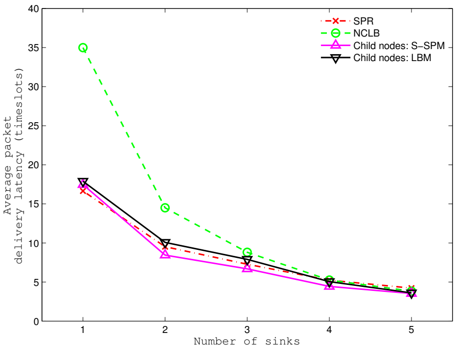
3.4.4 Shortest Path Relaxation Performance

By using basic shortest path routing nodes forward data always to a neighbor closer to the sink. This is the easiest method and guarantees a loop free routing tree. However, bottlenecks in the tree – congested or nearly depleted nodes – cannot always be avoided by this approach. Simulation results in Figure 3.11 obtained from the runs on the asymmetric cluster network topologies to show the effect of shortest path relaxation on the networking performance. Routing metrics *buffer* and *energy level* are taken into account, because they can benefit directly the best from the shortest path relaxation. Both metrics are simulated in S-SPM and LBM. Results with shortest path relaxation are colored yellow and without shortest path relaxation are colored green in the figure.

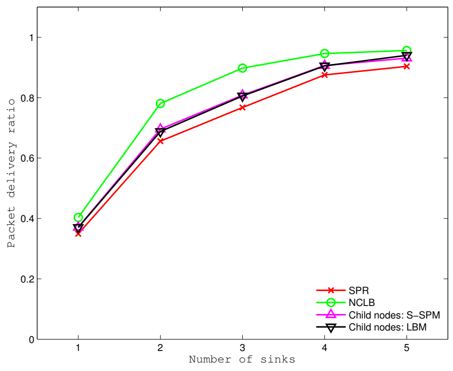
In Figure 3.11(a), standard deviation of sink load decreases a bit when we apply shortest path relaxation combined with routing metric *buffer*. The routing metric *energy level* with shortest path relaxation achieves a better balancing than *energy level* without relaxation.



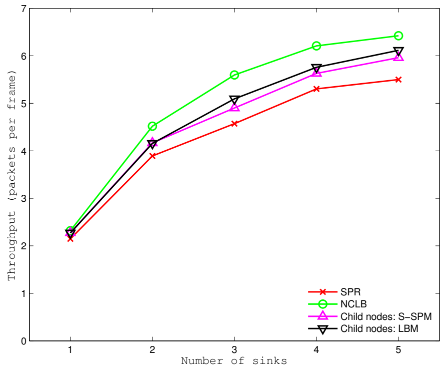
(a) Load Balance



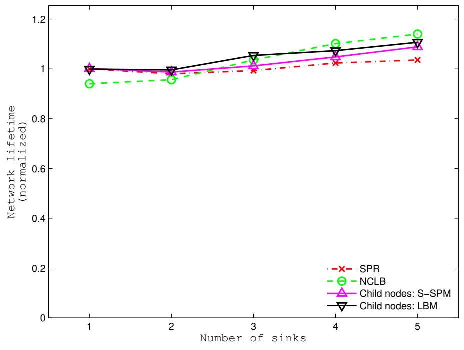
(b) Latency



(c) PDR

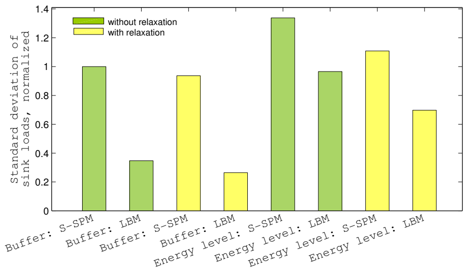


(d) Throughput

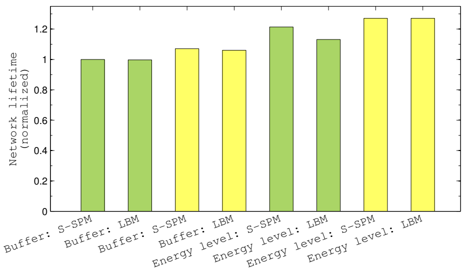


(e) Network Lifetime

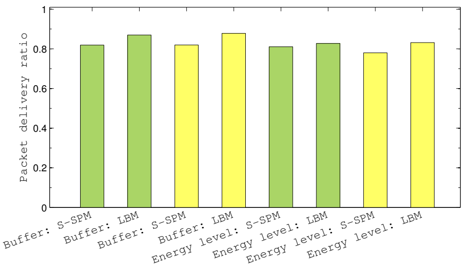
Figure 3.10: *Effects of multiple sinks*



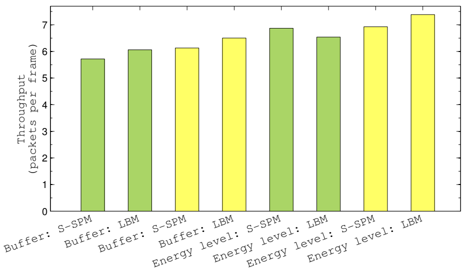
(a) Load Balance



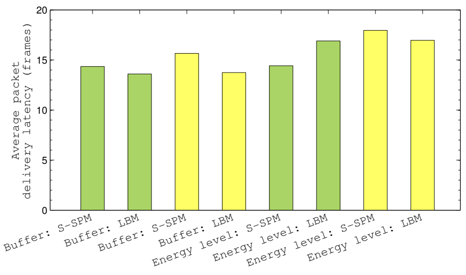
(b) Network Lifetime



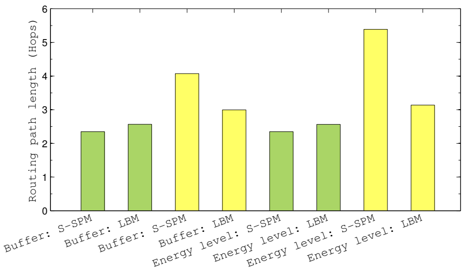
(c) PDR



(d) Throughput



(e) Latency



(f) Path Length

Figure 3.11: *Effects of shortest path relaxation*

When we compare the metrics *buffer* and *energy level*, *buffer* gives better results in terms of load balancing. Network lifetime also benefits from shortest path relaxation for both metrics in Figure 3.11(b). The effect of relaxation on delivery ratio is not much as shown in Figure 3.11(c). However, throughput increases with the shortest path relaxation approach in Figure 3.11(d). As expected, latency and average path length (i.e. number of hops) between sinks and sources increase when we apply shortest path relaxation in Figures 3.11(e) and 3.11(f), respectively.

3.4.5 Discussion of Simulation Results

With all the collected simulations results, it is clearly seen that the load balancing mechanism of LBM does balance the load more equally over all the sinks in the network. However, in the random network topologies, this does not have the expected positive effect on the other performance metrics. In the multi-sink simulations, LBM performs more or less equal to S-SPM. As excepted, routing metrics *buffer* and *network lifetime* are able to achieve a fair latency and network lifetime. In a regular topology with two clusters of different sizes, such as the asymmetric clusters topology, the benefits of load balancing are much more obvious. Latency, PDR, and throughput are between 10% and 40% better. When looking at application targets, the conclusion can be drawn that routing metric *buffer* leads to the lowest latency and the highest PDR, while routing metric *energy level* results in the highest throughput and the longest network lifetime. Although the load in the network is more balanced using balancing mode, we have seen that latency, PDR and throughput do not reflect this. In what follows we list the reasons of this behavior:

– Incidentally created loops cause temporary extra latency (until the loop is detected and broken).

– Longer path lengths due to better balancing cause extra latency.

– Sinks are in most cases not the bottleneck in the network, but congestion occurs sooner in nodes around the sinks – the top-level nodes. Indeed, a sink generally has much more bandwidth to the end-user (network) and is able to process nodes faster at its buffer. Therefore, balancing the load over the sinks does not always lead to a better performance. Balance between the branches of a sinks may be more crucial.

– Local bottlenecks generated due to irregular structure of the random topology networks have more influence on the performance than the load on the sinks in some WSNs.

– Bandwidth distribution of LMAC can be problematic. The collision free scheduling technique of TDMA-based LMAC protocol has some drawback in fixed but reduced bandwidth per node. Every node can send only one packet per frame, no matter if it has many more packets to send – when the node is congested – or there is no packets in its message queue. This increases the negative effect of bottlenecks on the performance of the protocol. This also limits the maximum throughput in the network where the maximum throughput is equal to the number of top-level neighbors of the sinks. Contention-based protocols like CSMA, might be better in reducing congestion, assuming the density of nodes is not too high.

The centralized NCLB algorithm performs in almost all cases better than the distributed P-NLB. Surprisingly, in some cases P-NLB is still able to outperform NCLB, for instance, LBM outperforms NCLB in most performance metrics in the asymmetric cluster topology simulations. The higher latency in NCLB is caused by the (much) longer routings path created by the balancing mechanism of NCLB. The main source of the better performance of NCLB is the cardinality of all top-level branches of the routing trees in the networks. LBM tries the balance the load of all the whole routing trees (clusters) in the network, while NCLB tries to balance the load of each top-level branch in the routing trees. Since the sinks have better processing capabilities than common sensor nodes, the sinks, themselves are not the critical points in many cases, but the top-level nodes are. Therefore, balancing the load in those top-level branches proves to be more effective. On the other hand, this is a much more harder task to achieve and thus may require a centralized method for optimal solution.

3.5 Conclusion

We have investigated the characteristics and problems of static multi-sink wireless sensor networks. We have presented a routing strategy called P-NLB for large-scale multi-sink WSNs, which uses global clustering with inter-cluster load balancing technique in combination with local metric-based routing for optimized routing tree building. On the global level, information about cluster sizes in the network is gathered by sinks and distributed to the sensors. Its distributed approach, in which each node autonomously decides what the best next hop node is, results in very low communication overhead due to the use of cross-layer information of the MAC layer and flexibility of the routing trees. Except for one-time broadcasting for detecting initial cluster sizes, which is a part of the LMAC setup phase, only local information exchange is used, making it very scalable to large scale sensor networks.

Simulations show that the load balancing mechanism of P-NLB uniformly distributes the load efficiently over the sinks in the network. In random network topologies this results in a higher latency, caused by longer routing paths. Packet delivery ratio does not always benefit from balancing the load; LBM gains the most advantage in comparison with shortest path mode, when the initial difference between clusters' sizes increases. Both routing modes of P-NLB outperform SPR in all simulations. NCLB achieves the highest performance in most of the simulations, which is not a surprise since it is a centralized approach. Evaluation of the four defined routing metrics show that using routing metric *buffer* leads to the lowest latency and highest PDR. When the application target is a long network lifetime or high throughput, using routing metric *energy level* leads to the best results.

In this chapter, we have considered statically deployed multiple sinks and sensors as a starting point to better understand the problems and challenges in static multi-sink WSNs. Top-level nodes, especially, one-hop neighbors of static sinks may suffer from congestion and quick energy depletion since they process more packets than lower-level nodes. Therefore, load balancing is more critical in static environments because sinks are always connected to same one-hop neighbors. Generally speaking, giving sinks the ability to move or using mobile sensors as relay nodes can help avoid this bottleneck problem. In the next chapters, we focus on the mobility aspects in WSNs and study on query and data dissemination protocols supporting mobility. The results obtained in this chapter can be combined with the protocols presented in the following chapters. For instance, metrics *buffer* and *energy level* can be used in the multi-sink partitioning and tree construction phases of the query dissemination protocol presented in Chapter 4.

CHAPTER IV *

Query Dissemination in Multi-Sink Mobile Wireless Sensor Networks

In order to efficiently deal with location dependent messages in multi-sink wireless sensor networks (WSNs), it is key that the network informs sinks what geographical area is covered by which sink. The sinks are then able to efficiently route messages which are only valid in particular regions of the deployment. The algorithms proposed in this chapter combine coverage area reporting and geographical routing of location dependent messages such as queries. We assume that the network is composed of static sinks and static or mobile sensor nodes. We propose a mechanism in which the WSN provides up-to-date coverage areas to sinks, and then sensors use the collected information on coverage areas for efficient routing of queries. The initial step concerns the construction of a structure consisting of routing trees, one for each sink, and coverage areas of these trees and their branches. The latter step focuses on routing of messages injected by sinks to sensor nodes in the region of interest, which is called geocasting. The periodic construction of the routing structure and geocasting of queries are discussed in the first part of the chapter. Also, geocasting of queries is extensively evaluated in both static and mobile scenarios in the first part. In the second part of the chapter, we focus on handling node mobility to achieve maintenance of the routing structure by avoiding periodic global updates in the network. We discuss what is a better method for updating the routing structure to handle mobility efficiently in tree-based geocasting of queries: periodic global updates initiated from sinks or local updates triggered by mobile sensors. Simulation results show that local updates perform very well in terms of query delivery ratio. It also is more energy efficient than global updating in networks having medium mobility rate and speed, independent of the size of the network.

*This chapter combines the following three publications: "Combined Coverage Area Reporting and Geographical Routing in Wireless Sensor-Actuator Networks for Cooperating with Unmanned Aerial Vehicles" In the Proceedings of 3rd ERCIM Workshop On eMobility, May 2009 [11], "Geo-casting of Queries Combined with Coverage Area Reporting for Wireless Sensor Networks" In Ad Hoc Networks Journal, Elsevier, Submitted in July 2010 [9], and "On Mobility Management in Multi-sink Sensor Networks for Geocasting of Queries" In Sensors Journal, MDPI, Submitted in May 2011 [7].

4.1 Introduction

A typical way of extracting information from a sensor network is to disseminate queries from sink nodes to sensor nodes, asking them to send data which has the properties specified in the queries. The main consideration in designing query dissemination algorithms is to efficiently forward queries from sink nodes to sensor nodes. If a query is only valid in particular regions of the deployment, it is important to determine which sinks need to inject the query to reach all sensor nodes in the area of interest. The simplest approach to deliver a message to all nodes within a geographical region (i.e. geocasting) is *simple flooding* of the message from all sinks in the whole network, irrespective of the destination region and coverage areas of the sinks. Since simple flooding is not an efficient approach in terms of communication overhead, another class of geocasting approaches, called *directed flooding*, has been proposed [116] to limit message overhead and network congestion by defining forwarding zone, which comprises a subset of all nodes in the network. To employ directed flooding in a multi-sink sensor network, the network should inform sinks what geographical area is covered by which sink. This requires partitioning of the network between different sinks. Multi-sink partitioning is generally performed using energy-aware or any other metric-based route selection schemes.

In the first part of this chapter, we design and evaluate a geocasting protocol which uses forwarding zones defined by local coverage areas of sensors and sinks in a tree-based network. The local coverage area of a node is the convex hull of the subtree rooted at this node. For the distributed construction of local coverage areas along the routing tree, every parent node receives the convex hull information from all its child nodes and merges them together with the addition of the parent node itself. This procedure constructs a new composite local coverage area, which is forwarded further up the tree until it reaches to the sink. The routing trees combined with local coverage areas (i.e. convex hulls) of sensors and sinks are called *geocast structure*. The local convex hulls are used for efficient geocasting of queries to the areas of interest from the sinks. Figure 4.1 illustrates local convex hulls of a sink and sensor nodes. When the sink injects a location dependent query into the network, it first checks if there is an overlap between the area covered by the sensor network and the area specified in the query. If so, the sink forwards the query to its child nodes. In their turn, these nodes check if there is overlap between their local coverage areas and the specified area of the query. If a node finds an overlap, it again forwards the query to its child nodes. If not, the query is simply not propagated. In this way, the query is routed to the area where it needs to be executed.

In the second part of the chapter (starting from Section 4.7), we go one step further, handling mobility in multi-sink sensor networks for coverage area based geocasting of queries. Here, the sensor network under consideration has a hybrid network architecture composed of fixed and mobile sensor nodes. In such a WSN, it is very crucial to support mobility of nodes and keep the routing trees and local coverage areas up-to-date. There is need for a mechanism to associate/re-associate mobile nodes and their child nodes to new parent nodes and update local coverage areas of sensors and sinks.

Managing node mobility in a tree-based forwarding scheme creates an extra overhead in the network. The reduction of the message overhead and the overall energy consumption of the WSN is the foremost goal as well as reliability of geocasting of queries. Hence, we

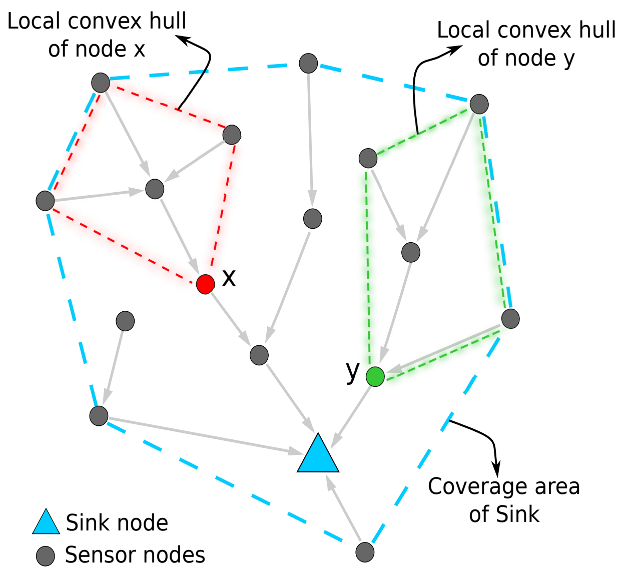


Figure 4.1: *Example network with coverage area (i.e. local convex hull stored in the sink) and a local convex hulls stored in sensor node x and y*

discuss what is the better method to handle mobility in tree-based routing of queries: Periodic *global updates* initiated by sinks or *local updates* triggered by mobile sensors. In the *global updating* mechanism, all sinks broadcast a ‘hello’ message to the whole network which result in a partitioning of the network between sinks and routing trees for each sink. After each node associates itself with a parent node, it transmits its local convex hull to its parent node. To handle mobility in this scheme, this procedure has to be repeated periodically. In the second part, we propose a local updating mechanism. The distinctive features of the proposed local updating solution are: (i) sending beacon packets from mobile sensor nodes instead of hello messages from every sink node, (ii) the use of proactive procedures to speed up the parent-child re-association and convex hull updating, and (iii) a reduced impact of the messaging overhead to manage node mobility by resorting to local updating procedures.

In the remainder of this chapter, we first discuss the related works in Section 4.2. We explain how we construct local coverage areas with the global updating mechanism in Section 4.3, and the geocasting of queries in Section 4.4. Section 4.5 discusses implementation aspects of the presented protocol. Performance evaluations of the convex hull based geocasting in static and mobile networks are given in Section 4.6. Section 4.7 describes the details of our local updating mechanism to efficiently handle sensor node mobility. We compare the performance of global and local updating mechanisms in Section 4.8. Finally, Section 4.9 draws the conclusions.

4.2 Related work

There are a number of related approaches in the area of query dissemination, geocasting and multi-sink partitioning in wireless sensor networks. In what follows we discuss some well known approaches briefly.

4.2.1 Query Dissemination

There are several approaches for disseminating query messages in a WSN. A flooding mechanism, which requires any intermediate receiver to rebroadcast a non-duplicated interest packet to all its neighbors, is the most commonly used technique. For example, in directed diffusion [85] one of the well-known query-based routing protocols in wireless sensor networks, a sink node initiates dissemination of an interest packet throughout the entire network by flooding. A node receiving the interest sets up a gradient which indicates from whom this interest message has previously been forwarded. Although some additional feature such as a gradient reinforcement has been proposed, the directed diffusion with such a flooding of interest messages obviously increases network traffic and leads to inefficient energy consumption on sensor nodes. There are some other proposals for query dissemination in WSNs such as minimum broadcast tree algorithms [171] or epidemic approaches like gossiping [60].

The approach we propose in this chapter differs from the above approaches because it uses position information to construct coverage areas. The convex hull based coverage area definitions [11] are used to scope the query dissemination between sinks and the target region. The protocols discussed in the next section are similar to our approach for using location information to perform dissemination.

4.2.2 Geocasting

A different routing strategy for wireless sensor networks is described in [180]: geographical routing. Instead of advertising an interest for data, or requesting to establish a route to a certain destination device, nodes use a routing technique based on node coordinates. Nodes are assumed to know their own position and the position of the destination node (i.e. the node where the message needs to be delivered). The idea is that nodes advertise data along with the coordinates where it must be delivered. Nodes closer to the destination node consider themselves candidates for relaying the message. In most of the geographical routing protocols such as Greedy Perimeter Stateless Routing (GPSR) [89], the packets are sent from source to a destination position. GPSR is a geographic routing protocol for wireless networks that works in two modes: greedy mode and perimeter mode. In greedy mode each node forwards the packet to the neighbor closest to the destination. When greedy forwarding is not possible, the packet switches to perimeter mode, where perimeter routing (face routing) is used to route around dead-ends until closer nodes to the destination are found. Face Routing [94, 38] routes packets along faces of planar network graphs by using simple right hand rule and proceeds along the line connecting the source and the sink. Although it guarantees to reach the destination, it does so with O(n) messages, where n is the number of network nodes, and a simple flooding algorithm already reaches the destination with O(n) messages. Also, it is not competitive with the shortest path algorithm in terms of cost depending on the number of hops between the source and the destination.

For some other scenarios like general position-based publish-and-subscribe services, it is also sufficient for some packets (e.g. queries) to reach any destination currently located in a given area, which is called geocasting. Yu et al. propose Geographical and Energy-Aware Routing (GEAR) algorithm [174], which shows how to broadcast a message to all the nodes in a target region. GEAR uses greedy forwarding to forward packets to the nodes that are progressively closer to the centroid of the target region, whilst trying to balance the energy consumption at the intermediate nodes. Once the message is delivered to the centroid of the target region, it uses restricted flooding, namely Recursive Geographic Forwarding, to broadcast the message to all remaining nodes in the given region. Instead of using geographical forwarding, GeoTORA [92] uses a unicast (ad-hoc) routing protocol (TORA [128]) to deliver the packet to the region and then floods within the region.

There are some other protocols based on window spanning infrastructure (WSI) for routing to the specified message window (i.e. destination region). In this approach, the message first is forwarded towards the message window by an end-to-end routing protocol. Once the message reaches the window, an infrastructure within the message window is built along with the message propagation. The method in [50] uses a Greedy technique to find a routing path from message originator to a node N_c located at the center of the messages spatial window. This first part of the routing is similar with the approach used in GEAR. For the routing inside the window, the framework proposed in [50] uses two different approaches namely WinFlood and WinDepth. The WinFlood algorithm consists of a constrained parallel flooding, where a node broadcasts the message to its neighbors only if its own location is inside the messages spatial window. The alternative solution, WinDepth, is based on depth first search policy.

As we have seen from the related works given above, the first step of geocasting is generally based on Greedy approach which cannot guarantee that a routing path to a node in the messages spatial window will be found. Stojmenovic [153] reviews the existing approaches for message delivery to a destination region, i.e. geocasting. Three approaches that guarantee delivery in static sensor networks are discussed in detail: (i) face traversal scheme based on depth-first search of the face tree, (ii) traversal of all faces that intersect the border of the geocasting region, and (iii) entrance zone (i.e. the set of points that are at smaller distance than the transmission radius R from the destination region) multicasting-based geocasting. These algorithms mainly solve the routing hole problem in sparse networks in order to guarantee the delivery of messages to the target region. However, face traversal has considerable communication overhead as we discussed previously, so these approaches cause unnecessary overhead in dense networks. An adaptation to traversal of faces intersecting the target region (called $GFPG^*$), which achieves delivery guarantee in sparse networks and reduces the additional overhead of face traversal scheme in dense networks, is discussed in [146]. In the $GFPG^*$ algorithm, each node inside the geocasting region divides it radio range into four equal partitions. If there is at least one neighbor in each partition, it is assumed that there is no gap around this node. Thus, this node will not send perimeter packets (i.e. initiate face traversal) and will send only the geocast message inside the target region. If a node has no neighbor in a partition, it enters the perimeter mode and uses right-hand rule to send perimeter packets.

The geocast routing protocols discussed above are non-flooding based approaches, meaning other routing protocols are used to reach the target region instead of flooding, e.g. greedy forwarding, ad-hoc routing. Regional flooding may still be used inside the target region. The authors in [116] also discuss *directed flooding* based geocast routing protocols. Directed flooding tries to limit the message overhead and network congestion of naive flooding by defining a forwarding zone, which consists of a subset of all network nodes. The forwarding zone (e.g. rectangle, cone) includes at least the sender of the geocast message and the target region of the message. It should also include a routing path between source node and target

Table 4.1: *Comparison of geocasting protocols (TR: Target Region, FZ: Forwarding Zone)*

Protocol	Path Strategy	Routing towards TR	Routing inside TR	FZ
GEAR (2001)	Unicast	Greedy Forwarding	Flooding	-
GeoTORA (2003)	Unicast	TORA ad-hoc routing	Flooding	-
WSI (2004)	Unicast	FullFlood/GreedyDF	WinFlood/WinDepth	-
GFPG* (2006)	Unicast	Greedy Forwarding wih face routing	Traversal of faces intersecting TR	-
LAR (1998)	Multicast	Directed Flooding	Regional Flooding	Rectangle
GeoGRID (2000)	Multicast	Directed Flooding	Regional Flooding	Rectangle
Voronoi (2003)	Multicast	Directed Flooding	Regional Flooding	Polygon

region. Otherwise, protocols either have to increase the size of the forwarding zone or fall back to simple flooding. An intermediate node forwards a message only if it belongs to the forwarding zone. Directed flooding based geocast protocols [93, 154, 104] differ in how they define the forwarding zone. In Location-Aided Routing (LAR) [93], the forwarding zone is the smallest rectangle that includes the sender node and the target region. In [154], the network is partitioned using the Voronoi diagram (i.e. forwarding zone) concept and each node forwards the packet to the neighbors whose Voronoi partitions (as seen by the forwarding node) intersect with the geocast region. The idea is to forward to a neighbor only if it is progressively closer to the target region. GeoGRID [104] partitions the network into logical grid cells and a single elected node close to the center of each grid cell is responsible for propagating geocast packets to neighboring cells.

Recently there are also proposals for geocasting of a message to several geocast regions. The authors in [36] combine clustering and multi-geocasting for delivery guarantee to multiple target regions in WSN. In this work we assume that each query packet specifies only one target region. We do not consider multiple target regions (i.e. multi-geocasting) for a single query packet. It is assumed that when a sink node needs to send the same query to different target regions, it has to generate separate query packets for each target region.

Table 4.1 presents a comparison of all discussed geocasting protocols. The main differences between the protocols are observed either in the first phase, which is flooding based or non-flooding based for routing towards target region, or in the second phase that is the routing inside the specified message window. However, our approach uses a different technique, which does not make a distinction between the two phases. Coverage area descriptions are used in the first phase of routing to forward the packets from source to the given area. It is again the coverage area description that is used in the second phase, transmitting packets to nodes inside the target region. In this chapter, we consider the query dissemination problem in a tree-based data collection and dissemination network. The approaches discussed above are not specifically designed for tree-based networks. Geographical approaches might either fail at dead-ends formed in random networks topologies or propose very complex solutions and require the computation of planar subgraphs of the connectivity graph to tackle routing holes. On the other hand, tree-based approaches work well for random topologies with potential empty areas (i.e. holes) in connected networks. By using convex hull based coverage area definitions and performing local updates, our approach provides support for sensor mobility. Moreover, it provides a simpler solution to the problem compared to other geocasting protocols. Since tree-based routing stores connectivity information in the routing tree, there is no need for a separate hole avoidance mechanism.

In the performance evaluations we compare our approach with GEAR [174]. All the non-flooding based geocasting approaches take GEAR as a basis. The other group of geocasting approaches, based on directed flooding, perform worse compared to GEAR[*] [116]: GEAR achieves higher delivery success ratio and lower message overhead than the other directed flooding protocols with rectangle and cone forwarding zones.

4.2.3 Multi-sink partitioning in WSNs

As discussed in Chapter 2.1, multiple sinks (multi-sink) appear as a solution for large scale networks [53, 145]. However, deploying more sink nodes in a WSN brings another question: How to partition a sensor network among multiple sinks? The simplest way is for each sink to propagate a message to the whole network to form partitions. However, global flooding from each sink is redundant and costly. To reduce message redundancy some flooding scoping techniques such as TTL scoping or geographical scoping are used. TTL scoping defines a time-to-live for messages disseminated from sinks. In geographical scoping, a node only re-forwards a message if the message came from the closest sink, where ‘closest’ means shortest euclidean distance. One of the geographical scoping methods proposed in [63] is Voronoi decomposition, where scoping decision is entirely distributed. This method describes Voronoi clusters to bound the propagation of messages from different sinks. Each node only rebroadcasts flood messages coming from closest sink, where ‘closest’ depends on the underlying distance metric. With this approach, flooding overhead remains constant independently of the number of sinks.

The main focus of this chapter is how to achieve efficient geocasting of queries and how to maintain the routing trees together with local coverage areas when we have mobile nodes in the network. Our protocol can be combined with any multi-sink partitioning technique and any tree-formation metric, e.g. energy level, number of child nodes, to create routing trees (see Chapter 3). In our implementation we have used shortest path routing metric [62] to create the trees, and Voronoi decomposition based on hop-count metric for multi-sink partitioning. In the following sections we give the details of our geocasting protocol based on local coverage area descriptions and describe the supporting approach, local updates for handling sensor mobility.

4.3 Coverage area reporting

In this section we discuss the design for distributed coverage area construction and reporting. Throughout this work, it is assumed that one or more sinks are deployed within the wireless sensor network and each sensor node is logically grouped with only one sink based on a given metric, e.g. shortest path. Basically, the routing strategy of the wireless sensor network determines which node reports to which sink. This work assumes that a routing tree is present to route the data efficiently towards a selected group sink e.g. the work in [99].

[*]GEAR protocol is called as Unicast Routing with Area Delivery (URAD) in [116]. URAD identifies a protocol class having two phases: the unicast (greedy) forwarding from the initial sender until the first node inside the target region is reached, and the flooding inside the target region.

The construction of local coverage areas starts after the routing trees are created. In the distributed approach of establishing a description of WSN coverage area per sink, nodes need to keep track of partial information of the coverage area. In this way, sinks are efficiently informed of the coverage areas, while the amount of information each node needs to store, transmit and receive is limited. Throughout this work, the limitations of sensor nodes in terms of resources (Section 4.5) are a driving force behind design choices.

4.3.1 Constructing local coverage areas

We assume that each node in the wireless sensor network has the ability to obtain an estimate of its position. This can be either by localization mechanisms [33, 58, 78, 80, 124], GPS or by other means (e.g. [69]). Whenever a node publishes information, it is augmented with the current position of the node.

An overview of the presented approach of establishing coverage areas is depicted in Figure 4.1. Nodes keep track of coordinates that are explicitly transmitted. Using the received coordinate information, a node creates its local coverage area. By the term *coverage area*, we understand the geographical area in which the sensor nodes are deployed. In this work, coverage areas are represented by convex hulls of the sensor node locations[†]. Periodically, the local convex hull is transmitted to the parent node that merges the received convex hull with its local coverage area. A parent node maintains a convex hull that envelopes the node itself and all its descendant nodes in the tree (see the local convex hull of node x in Figure 4.1). The coverage area of a sink is the convex hull of all the sensor nodes served by this sink (see Figure 4.1).

With this ‘crude’ coverage area description detail is lost of e.g. holes in the wireless sensor network deployment. Many geographical routing protocols need to take special precautions to ensure that messages are not stuck at holes in the deployment. However, the coverage area descriptions are built based on connectivity information captured in routing trees; therefore, the holes are implicitly avoided.

Each node keeps a convex hull that encloses all nodes in its subtree (see Figure 4.1). In the following, we discuss how a node constructs a local coverage area that describes the area covered by the node itself, its child nodes and other descendant nodes. To create a local coverage area description, a node inspects all messages that ‘flow’ through the node towards the designated sink. The location information inside these messages is used as input for the construction algorithm.

Let $\mathbf{C_0} = \{c_0, c_1, \ldots, c_s\}$ be a set of locations of s nodes, where each location c is a two dimensional coordinate $(c^{(x)}, c^{(y)})$. Let the function $\mathcal{CH}(\mathbf{C_0}) = \mathbf{H}$ create a minimal (ordered) set of coordinates $\mathbf{H} \subseteq \mathbf{C_0}$ that envelops the coordinates in set $\mathbf{C_0}$. $\mathbf{H}$ is called the convex hull of the coordinate set $\mathbf{C}_0$. We assume that the coordinates in $\mathbf{H}$ are ordered such that the convex hull encompasses the coordinate set $\mathbf{C_0}$ counter clockwise. We denote $|\mathbf{H}|$ as the number of elements in the set $\mathbf{H}$. Note that $|\mathbf{H}| \leq |\mathbf{C}_0|$.

Many methods are described in literature that transform a set of coordinates to a convex hull e.g. [55]. Typically, these algorithms operate on a set of coordinates and produce a convex hull, but most of them do not consider addition of coordinates once the convex hull has been created. In the following, we present an algorithm that constructs and maintains the

[†]Convex hull of a node is the local coverage area description of this node. These terms are used interchangeably in the chapter.

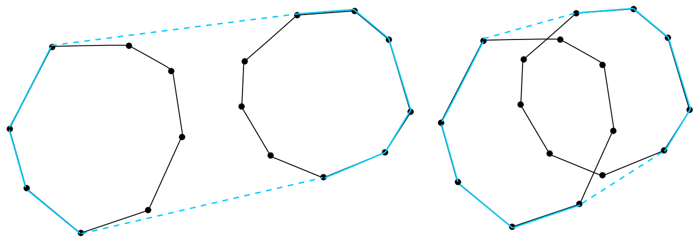


Figure 4.2: *Merging convex hulls with rotating calipers method: (a) non-overlapping convex hulls and (b) overlapping convex hulls*

(local) convex hull of a node when periodically transmitted local convex hulls are received by the node. In fact, to construct convex hulls along the routing tree, we need an algorithm which implements a merge function $\mathcal{CH}(\mathbf{H}, \mathbf{C}) = \mathbf{H}'$, where $\mathbf{C}$ can be ***(i)*** a single coordinate, ***(ii)*** a set of coordinates, or ***(iii)*** a coordinates set representing a convex hull, with $\mathbf{H}' \subseteq \mathbf{C} \cup \mathbf{H}$.

The simplest and the most efficient way to achieve merging of two convex hulls (i.e. case ***(iii)***) is applying the rotating calipers based algorithm [147, 159] locally in each parent node. Figure 4.2 shows merging of convex hulls. The merged hull consists of convex chains belonging to the polygons (shown is solid blue lines in the figure), joined by bridges between the polygons (shown in dashed blue lines in the figure). Rotating Calipers are used to find the bridges between two convex polygons. The main advantage of rotating calipers method is that it involves no backtracking, and the polygons can intersect whereas other algorithms require the polygons to be disjoint. The algorithm has a runtime complexity of $O(n \log n)$ and a space complexity of $O(n)$ [159]. However, rotating calipers method cannot handle the merging of a convex hull with a single coordinate (i.e. case ***(i)***) or a set of coordinates (i.e. cases ***(ii)***). In the following we present an algorithm that performs incremental construction of convex hulls, which covers for all the three cases discussed above.

For incremental construction of a convex hull, we need to check if the new coordinate is inside the convex hull. This is done by checking whether the coordinate is always on the left side of each edge of the convex hull, where the coordinates of the convex hull are in counter clockwise order. Let $\overrightarrow{h_k h_{k+1}}$ be the vector connecting location h_k with h_{k+1} of the convex hull $\mathbf{H}$. When the indices are larger than the size of the set e.g. when $k + 1 > |\mathbf{H}|$, the modulo with the set size is meant. To determine if a coordinate c is on the left side of a vector, we make use of the right hand-rule, by checking the orientation of the cross product $\overrightarrow{h_k h_{k+1}} \times \overrightarrow{h_k c}$. In the two-dimensional case, the cross product $\overrightarrow{h_k h_{k+1}} \times \overrightarrow{h_k c}$ is equivalent to

$$d = (h_k^{(y)} - h_{k+1}^{(y)})c^{(x)} + (h_{k+1}^{(x)} - h_k^{(x)})c^{(y)} + h_k^{(x)} h_{k+1}^{(y)} - h_k^{(y)} h_{k+1}^{(x)} \tag{4.1}$$

The coordinate c is on the left of line segment $\overrightarrow{h_k h_{k+1}}$ if the result of Equation (4.1) is positive: $d > 0$.

Let p_i to be the position of node i in the wireless sensor network, and $\mathbf{H_i}$ be the convex hull representing the (local) coverage area description of the wireless sensor node or sink i. The coordinate set $\mathbf{H_i}$ is always ordered such that it describes the convex hull counter clockwise. Initially, $\mathbf{H_i} = \{p_i\}$ contains the coordinate of the node itself. However, during the update process described below, the coordinate of the node itself might be removed from $\mathbf{H_i}$.

Let $\mathbf{C}$ be the set of coordinates that a node or sink receives ($\mathbf{C}$ is either a single coordinate which is extracted from a sensor reading flowing through the node, or a received convex hull from a child node). Per coordinate in the set $\mathbf{C}$ the following procedure is executed:

1. Define c_j as current coordinate to investigate from the set $\mathbf{C}$ $(0 \leq j \leq |\mathbf{C}| - 1)$. If this coordinate is already present in the set $\mathbf{H_i}$, move on to the next coordinate. We investigate per coordinate if it is inside $\mathbf{H_i}$. If not, $\mathbf{H_i}$ is adjusted such that it envelops the coordinate as well.

2. Let $n = |\mathbf{H_i}|$ be the number of coordinates in the local convex hull:

   - **One coordinate** ($n = 1$) — Add the coordinate to $\mathbf{H_i}$ and order the coordinates such that the coordinate with lowest y-value is first in the set.
   - **Two coordinates** ($n = 2$) — Use Equation (4.1) to check if c_j is on the left of the line segment $\overrightarrow{h_0h_1}$. If so, put the coordinate at the third position in the convex hull $\mathbf{H_i}$, otherwise insert the coordinate between h_0 and h_1 in $\mathbf{H_i}$.
   - **More coordinates** ($n > 2$) — Check for each line segment $\overrightarrow{h_0h_1}$, $\overrightarrow{h_1h_2}$, …, $\overrightarrow{h_{n-1}h_n}$, $\overrightarrow{h_nh_0}$ if the coordinate c_j is on the left of the line segment. If so, the coordinate is enveloped by the convex hull $\mathbf{H_i}$; continue with the next coordinate.

     If c_j is not on the left of a line segment, then record the starting coordinate of the line segment as begin point b. Continue with the next line segments until c_j is left of the line again. Remove all coordinates from b until the current line segment and insert c_j instead.

The above procedure is applied when a node receives a packet augmented with position information or when local convex hulls are explicitly propagated from child nodes. We use the definitions *parent node* and *child node* to indicate node positions in the routing tree. Next, we describe tasks that nodes need to execute periodically to keep their local coverage area up to date in dynamic networks.

To keep the routing tree and local convex hulls up-to-date, a *global updating* mechanism should be executed periodically. The periodic *global updates* consist of two phases: (i) *helloing from sinks* for tree building and multi-sink network partitioning, and (ii) *convex hull forwarding* to the parent nodes for completing the structure with local convex hulls of sensors and sinks. In the first phase, each sink repeats broadcasting 'hello' packets every f seconds (i.e. frequency) to determine the paths to every node in the network and constructs a routing tree based on the given metric. A child node has selected a parent node as intermediate node in order to get messages towards a sink after helloing process. In this work, nodes store the logical address of their parent node. In the second phase each node sends a `HELLO_TO_PARENT` message including its convex hull to its parent. Every parent node receiving a convex hull from all its child nodes merges the received convex hulls with its current local convex hull. This procedure constructs a new composite convex hull, which is forwarded further up in the tree until reaching to the sink. This allows the construction of a hierarchy of convex hulls. The distributed procedure requires $O(1)$ messages per sensor node assuming that each parent node waits until it gets the convex hulls of all its child nodes, $O(dm)$ space and runs in $O(dm)$ time, where d is the maximum node degree in the network and m is the maximum number of nodes in a convex polygon. Optionally, the convex hull is reduced using some form of compressing before transmitting (in order to limit the memory usage by the algorithm and the energy consumption by reducing the size of transmitted/received coordinate list) as explained in Section 4.3.3.

4.3.2 Removing coordinates from local coverage areas

Due to dynamics in network topology, the local convex hull stored in a node can contain coordinates that do no longer reflect the actual coverage area of the node, its children and other descendants e.g. this might be the case when a node dies/fails or nodes are mobile. To keep the local convex hull accurate, a time out mechanism is applied to remove old coordinates from the local convex hull. Nodes store a timestamp for each individual coordinate in their local convex hull $\mathbf{H_i}$. The timestamp of a particular coordinate is reset when a node receives a message containing the coordinate. But when a coordinate has not been reinforced within the time out interval, it is removed from the local convex hull and is therefore also not propagated to the parent node. The time out information is never propagated to parent nodes.

A suitable *timeout interval* needs to be determined according to the level of mobility in the network. However, it must not be shorter than the interval at which nodes produce and send their convex hulls to their parents, otherwise coordinates are removed from the local convex hulls before they are reinforced. If topology changes are frequent, the *timeout interval* should be short to ensure up-to-date coverage area descriptions. Also, the local convex hull needs to be transmitted to parent nodes at least once per time-out interval. A timer relating to the local convex hull update interval is employed.

$$Timeout = Interval + T_D \tag{4.2}$$

where T_D is a compensation value with respect to additional delay.

Periodically i.e. once per timeout interval, a node applies the algorithm described in Section 4.3 to check if its own position p_i must be added to the local coverage area description. This action also ensures that a potential time out on the own coordinate is prevented.

In some cases, significant changes in the routing tree may be used as trigger to recreate all local coverage areas in the network. A significant degradation in the dissemination performance can be interpreted as a significant change in the routing tree. In this case, a *global updating* procedure has to be executed in the network.

4.3.3 Compression of coverage area descriptions

The proposed mechanism for distributed coverage area reporting requires that nodes (periodically) propagate the convex hull that describes the local coverage area to parent nodes. Obviously, message sizes grow with the number of coordinates that are part of the convex hull. Consequently, more accurate, but larger coverage area descriptions result in higher energy expenditure of the nodes. Therefore, compression (i.e. approximation of the convex hull with a smaller coordinate set) is an attractive option to limit resource consumption, such as energy and bandwidth. It is important to note that a convex hull is already a minimum set by itself.

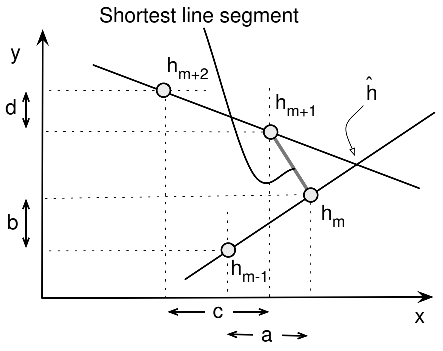


Figure 4.3: *Compression of the convex hull removes short line segments from* **H** *by adding coordinates at the intersection of segments before and after the short segments*

The compression algorithm accepts as input a convex hull and a maximum convex hull size $n_{max} > 3$. Until the convex hull has been reduced to maximum size n_{max}, the algorithm finds two coordinates h_m and h_{m+1} which represent the shortest line segment in the convex hull with $r_m \neq 0$ (Equation 4.5). These two coordinates are then removed from the convex hull **H** and are replaced with one coordinate $\widehat{h}$, such that h_m and h_{m+1} are both on the line segments $\overrightarrow{h_{m-1}\widehat{h}}$ and $\overrightarrow{\widehat{h}h_{m+2}}$, respectively (Figure 4.3). The coordinate $\widehat{h}$ is positioned at the intersection of the line passing through h_{m-1}, h_m and the line through h_{m+1}, h_{m+2}. It is calculated as follows [24]:

$$\widehat{h}^{(x)} = \frac{1}{r_m} \begin{vmatrix} h^{(x)}_{m-1}h^{(y)}_m - h^{(y)}_{m-1}h^{(x)}_m & h^{(x)}_{m-1} - h^{(x)}_m \\ h^{(x)}_{m+1}h^{(y)}_{m+2} - h^{(y)}_{m+1}h^{(x)}_{m+2} & h^{(x)}_{m+1} - h^{(x)}_{m+2} \end{vmatrix} \tag{4.3}$$

$$\widehat{h}^{(y)} = \frac{1}{r_m} \begin{vmatrix} h^{(x)}_{m-1}h^{(y)}_m - h^{(y)}_{m-1}h^{(x)}_m & h^{(y)}_{m-1} - h^{(y)}_m \\ h^{(x)}_{m+1}h^{(y)}_{m+2} - h^{(y)}_{m+1}h^{(x)}_{m+2} & h^{(y)}_{m+1} - h^{(y)}_{m+2} \end{vmatrix} \tag{4.4}$$

with

$$r_m = \begin{vmatrix} h^{(x)}_{m-1} - h^{(x)}_m & h^{(y)}_{m-1} - h^{(y)}_m \\ h^{(x)}_{m+1} - h^{(x)}_{m+2} & h^{(y)}_{m+1} - h^{(y)}_{m+2} \end{vmatrix} \tag{4.5}$$

Note that $r_m = ad - bc$ according to the definitions in Figure 4.3. If the intersection point $\widehat{h}$ does not exist i.e. the two lines $\overrightarrow{h_{m-1}h_m}$ and $\overrightarrow{h_{m+1}h_{m+2}}$ are parallel and hence have equal slopes $b : a = d : c$, then follows $r_m = 0$. When this is the case for a shortest line segment, it is skipped by the compression algorithm. Consequently, convex hulls with $|\mathbf{H}| = 4$ having parallel opposite line segments, cannot be further reduced. However, a reduction to a triangle is possible in other cases. Hence, we limit $n_{max} \geq 3$.

Reduction is only applied when a copy of the local convex hull is forwarded to parent nodes. Nodes maintain the actual convex hull in memory to use detailed information for the geographical routing decisions (Section 4.4). Obviously, the larger the local coverage area descriptions, the more memory is consumed by the uncompressed convex hull, more processing is needed to apply the merging of coordinates (Section 4.3.1) and the energy expenditure of nodes will be larger. Therefore, it could be a trade-off to apply compression also to the local convex hull. However, when reduction is only applied on copies forwarded to parent nodes, the time out mechanism of coordinates remains functional without having to e.g. match coordinates to substituted coordinates. In any case, the parent node works with the compressed version. This implies that compressed local convex hulls need to be forwarded to parent nodes within the retention period of coordinates to ensure that substituted coordinates are not removed due to time out mechanism.

4.4 Geocasting based on local coverage areas

With the above described algorithms, the sinks are informed of a 'crude' description of their coverage area. Next, this information can be used to optimize handling of position dependent messages e.g. sinks can use the information whether a certain query is relevant for their coverage area. If not, the sink can decide to discard the query without inserting it in the WSN, which in the end saves energy and prolongs the lifetime of the wireless sensor network. In this section, the geographical routing of location dependent queries is discussed.

The proposed protocol based on the combination of coverage areas and geographical routing is called *GeoCHT*, where "CHT" (Convex Hulls Tree) stands for the structure composed of routing trees and convex hulls built on a tree, and "Geo" stands for the geographical routing the WSN performs using the structure. Sinks and sensor nodes implement identical functionality regarding the forwarding of queries.

First, we have a closer look at the structure of location dependent queries. We assume that these queries consist of two parts: (1) a description of the area in which the query must be executed, and (2) a command sequence (e.g. sensor types, sample rates, critical thresholds, aggregate functions etc). This work is mainly concerned with the first part of the query. Queries are always forwarded from parent nodes to child nodes to get delivered to an area that is specified in the query. Let $\mathbf{R} = \{r_0, r_1, \ldots, r_n\}$ be the coordinate set describing the region of interest extracted from the query, $\mathbf{H_i}$ the local coverage area description of node i and p_i the position of node i.

Upon receiving a query, a node analyses $\mathbf{R}$ and takes two decisions: (1) *execute decision* to find out if the node is within the region of interest and needs to execute the query and (2) *forward decision* to find out if the node has child nodes or further descendants in the region of interest. Both decisions use $\mathbf{R}$ as input together with p_i and $\mathbf{H_i}$, respectively (Figure 4.4).

4.4.1 Execute Decision

When a node receives a query, it decides if the query is valid for it and, if so, the query is stored and executed until it expires. The *execute decision* basically checks if the node that receives the query is inside the region of interest i.e. if point p_i is inside the polygon $\mathbf{R}$. The point-in-polygon problem is a well known problem in computational geometry and many solutions and implementations have been proposed [144]. Looking from a node implementation perspective, it is beneficial to make the assumption that the region of interest in the query $\mathbf{R}$ is a *convex* polygon. This assumption is not strictly necessary, but it reduces the complexity of the implementation because of properties of a convex polygon. When the area of interest cannot be captured with a convex polygon, we assume that multiple queries are generated

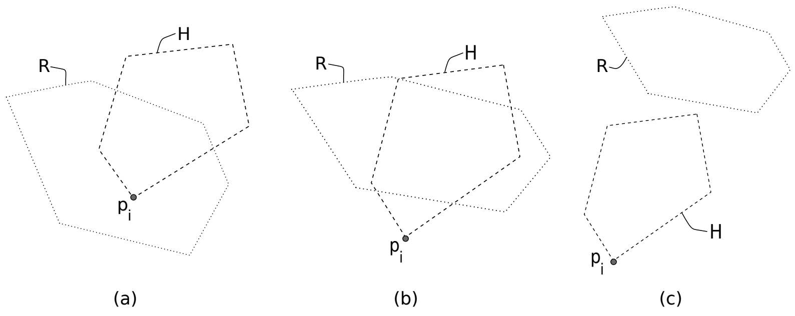


Figure 4.4: *Routing decisions: (a) node i executes query if p_i inside* **R**, *(b) node i forwards query to children if* $\mathbf{H_i}$ *overlaps with* **R**, *and (c) node i halts forwarding query to children if* $\mathbf{H_i}$ *and* **R** *are disjoint*

to cover the complete area e.g. according to the algorithms presented in [44, 72] to decompose polygons into multiple convex parts. This functionality can be realized e.g. within the different sinks, which are likely to be more computational powerful than sensor nodes.

To check if the coordinate p_i is inside a convex polygon **R**, node i needs to check for every line segment the coordinate p_i is left of the line segment. If so, the query needs to be executed (see Figure 4.4(a)). In the simulations we assume that the target region is a circle having a radius r_C. The node needs to check the distance between its coordinate p_i and the coordinate of the center of the area of interest. If the distance is smaller than r_C, the node should execute the query.

4.4.2 Forwarding Decision

With the forwarding decision a node determines if there *might* be child nodes or nodes further down the routing tree that are within the area of interest specified in the query. If there are, the node should forward the query to its child nodes, which in their turn decide if the query needs to be propagated.

When the coverage area is represented by a convex hull, a node A cannot determine with certainty that there is indeed any node in the subtree of A and within the polygon **R**, because detail on node positions are lost if nodes are located within the convex hull. However, a node is able to decide with certainty that further in its part of the routing tree no node is present within the region **R**. In the later case, the node does not forward the query and it consequently does not spend energy on transmitting the query and it saves resources from its child nodes.

The forwarding decision is taken based upon **R** extracted from the query and the local coverage area description $\mathbf{H_i}$. Checking if **R** overlaps with $\mathbf{H_i}$ (i.e. determining if two convex polygons intersect) can be easily performed in $O(\log(m+n))$ as described in [45], where m and n are the numbers of vertexes of the two polygons, respectively. We assume that the target region is a circle in our simulations. Checking if a circle **R** overlaps with $\mathbf{H_i}$ is done by "partially" checking the circle against edges [22], and against vertexes as shown in Algorithm 1. The first step is checking whether the center point of the circle is inside the convex polygon. This is the well-studied point-in-polygon problem [144]. If the center point of the circle is inside the polygon, the circular area of interest and the convex hull intersect.

Algorithm 1 Circle and Convex Polygon Collision Detection

```
1: Input: Polygon H, Circle R
2: Output: true or false
3: if H.contains(R.center) then
4:    return true;
5: end if
6: for all $edge_i \in$ edges of H do
7:    if distance(R.center, H.$edge_i$) < R.radius then
8:       return true;
9:    end if
10: end for
11: return false;
```

If not, the second step should be checking the distance between the center point of the circle and each edge of the convex polygon. If the distance between any of the edge of the convex hull and the center of the circle is shorter than the radius of the circle, than the circle and the convex hull intersect. If the area of interest and the local coverage area intersect, the node forwards the query (see Figure 4.4(b)). Otherwise, it halts the forwarding of the query (see Figure 4.4(c)).

4.4.3 Discussion

After routing tree construction every node knows its parent node in the tree, the sink it is connected to and the hop count from the sink. In a tree based routing protocol, there are two cases for parent-child relationship. The first case is where a parent node knows its child nodes and the other case is where it does not. In the first case, a parent node waits for the convex hulls of all its child nodes and after merging them, it forwards its complete local convex hull to its parent. This case can minimize the communication overhead for the transmission of queries or data, but requires more memory on the parent nodes and a large number of control messages for letting the parent nodes know their child nodes. In the second case, a heuristic has to be used to determine the delay until all the convex hulls of the child nodes are transmitted to the parent. This approach may result in a large delays for building convex hulls of tree branches. For a network which assumes that sensor nodes do not know their child nodes, a special multicast addressing that represents restricted flooding to child nodes of a node can be used for query forwarding (i.e. a node decides to receive a packet if it carries the multicast address and it originates from its parent node).

If also support for node mobility is required, a potential optimization step concerning this aspect is to keep track as parent node which area is covered by which children and only forward the query to relevant child nodes. This requires for every parent node additionally to store the convex hulls of its child nodes. Storing convex hulls of the child nodes can be useful for parent nodes to recalculate their convex hulls after a mobile child moves out of their communication range without having to request convex hull information of its children for every update. However, it must be noted that storing convex hulls of the child nodes requires nodes to store more information, makes the halt or forward decision more resource consuming, because it must be repeated for all stored local convex hulls of children, and the energy consumption of transmitting queries to more than one (i.e. unicasting instead of multicasting or broadcasting) will add considerably to the energy consumption of a node.

We leave these trade-offs to our future work and assume that parent nodes do not know their child nodes in the simulations. Therefore, for the parent node of a mobile node to remove the local convex hull of the mobile child moving out of its range and update the own convex hull, the parent has to re-request local convex hulls of its current child nodes.

4.5 Implementation aspects for resource-constrained sensor nodes

The presented algorithms to obtain coverage area descriptions and the geocasting of location dependent queries using coverage areas, have been designed with resource-constrained sensor nodes in mind. Typically, these platforms use microcontrollers running at 4 to 8 MHz with on-chip RAM and program memory. The non-volatile program memory ranges between 32 kB and 128 kB, while the volatile memory is considerably smaller, ranging from 2 kB to 10 kB. Usually the nodes can access external non-volatile memory to store arbitrary data. In this section, we dive into the implications it has to implement the presented algorithms on resource-constrained sensor nodes in terms of computational complexity and memory usage.

Let denote with s_c and s_{ct}, respectively the number of bytes required to store a single coordinate and a coordinate with time out information.

4.5.1 Computational resources

The computational complexity of the algorithms – discussed in the previous sections – have been well studied in literature. In [23], the computational complexity is discussed of incremental construction of a convex hull: $O(n^2)$ for the construction of the complete convex hull. Toussaint discusses in [159] the rotating calipers method of merging convex hulls with computational complexity $O(n \log n)$. The computational complexity of determining whether a point is in a convex polygon is $O(n)$ according to [83]. In [45], the complexity of determining if two convex hulls overlap is given as $O(\log (n + m))$.

4.5.2 Memory requirements

Let $|\mathbf{H_i}|$ be the size of the local coverage area of a node i. Obviously, the node requires $|\mathbf{H_i}|s_{ct}$ bytes to store its local convex hull, $|\mathbf{C}|s_c$ bytes to store temporarily coordinates received from a child node, where $|\mathbf{C}|$ is the number of coordinates in the coordinate set $\mathbf{C}$, and, at most, $|\mathbf{H_i}|s_c$ bytes to use as working copy of its local convex hull to create a compressed version. Obviously, storage is also required for a received query and its area of interest $\mathbf{R}$. We consider here only the temporary storage space to analyze the query i.e. $|\mathbf{R}|s_c$ bytes. After analyzing the query, we assume that it is either purged from memory or it will not account to memory consumption in the routing layer, since it is handled by a higher layer. In conclusion, the total memory requirements for the presented routing scheme are $|\mathbf{H_i}|(s_{ct} + s_c) + |\mathbf{C}|s_c + |\mathbf{R}|s_c$ bytes of which $|\mathbf{H_i}|s_{ct}$ bytes are permanently required.

4.6 Performance Evaluations

A comparative performance evaluation of GeoCHT versus GEAR [174] is presented in this section. In [116], the performance of different geocasting protocols is compared and GEAR shows the best performance over all the others. Therefore, we choose GEAR for our comparative performance evaluation. First, we define what metrics are used to compare both protocols in simulation. Then, the details of simulation setup are discussed. We compare the routing performance in different set-ups: static networks, mobile networks, high density and low density. The evaluation results of the different scenarios are presented and discussed.

4.6.1 Evaluation metrics

The geographical routing protocols are compared in terms of routing accuracy i.e. how well the proposed mechanisms deliver messages to the region of interest defined in a query and in terms of networking performance. Differently from GEAR, GeoCHT also provides a description of coverage area per sink in the network. This aspect is left out of the comparison, although it enables more efficient use of a WSN from an application point of view.

We define the following metrics to compare the protocols in terms of routing accuracy :

- Execution ratio (ER) – The ratio of nodes that are within the region of interest and execute the query to the total number of nodes within the region of interest. This metric measures how well the routing is able to deliver the query to the region of interest.
- False execution ratio (FER) – The ratio of the nodes that are outside the region of interest and execute the query to the total number of nodes outside the region of interest. Energy is wasted when queries are executed outside the region of interest. The false execution ratio measures this effect.
- False injection ratio (FIR) – The ratio of data sinks that inject the query while none of the nodes in its partition executes the query to the total number of data sinks. Since there is no partitioning in GEAR protocol, FIR is redefined for GEAR as the ratio between the number of data sinks that inject the query while none of the nodes inside the target region execute the query injected by these sinks and the total number of data sinks. Irrelevant query lead to higher energy expenditure in the WSN partition when injected. We measure this effect with the false injection ratio.

We define the following metrics to compare the protocols in terms of networking performance:

- Average query delay (AQD) – The total time elapsed between the query generation by a sink and its reception by a sensor node inside the target region, averaged over all sink-target node pairs.
- Network load (NL) – The total number of *query packets* that are sent from sinks and forwarded by the nodes in the network. If more messages need to be transmitted to reach the region of interest, the energy expenditure in the WSN is likely to increase. We measure this effect with network load.

4.6.2 Evaluation setup

We evaluate how our GeoCHT algorithm compares with GEAR, which also handles geographical routing of messages to a region of interest. The details of GEAR are given in Section 4.2. Table 4.2 presents the simulation parameters.

Table 4.2: *Simulation parameters*

Parameters	Values for GeoCHT	Values for GEAR
MAC protocol	IEEE 802.11 DCF	IEEE 802.11 DCF
Routing protocol for coverage area reporting	Shortest path routing	-
Routing protocol for queries	Convex hull based geocasting	Greedy forwarding Restricted flooding
Tranmission range	250 *m*	250 *m*
Target region shape	Circle - radius from 250 *m* to 800 *m*	Circle - radius from 250 *m* to 800 *m*
Mobility Model	Linear mobility	Linear mobility

Our simulations are based on the network simulator *NS-2* [15] version 2.33 as it is widely used for research in wireless sensor networks. *NS-2* allows us to run wireless simulations in realistic scenarios to validate the design choices of GeoCHT. We use 802.11 as the MAC protocol for our *NS* simulations. The simulation scenarios use network sizes from 100 to 1000 nodes, which are randomly distributed in square deployment areas with edge sizes varying from 1000*m* to 5700*m*. The nodes are configured to have 250*m* transmission range. We generate random topologies, however, only use connected network topologies (i.e. all nodes can reach each other in one or more hops).

For both GeoCHT and GEAR, the target region is set to a circle centered at the middle of the deployment area. The radius of the circular target region is selected between 250*m* and 800*m*, scaled according to network dimensions. The routing decisions of GeoCHT based on point in convex polygon (Section 4.4) have been adapted to point in circle to match the methodology used in GEAR to describe the region of interest. In the different scenarios, 3 to 30 sinks are randomly selected in each of the varying network sizes. For GeoCHT, sensor nodes send their convex hull definitions to their associated sink, via parent nodes selected by using shortest path routing.

In mobile simulation scenarios, node movement follows the linear mobility model. Node velocities are up to $20m/s$ (72 km/h), which includes walking/running person and vehicular movements. Each data point in the graphs are averaged over 50 simulation runs of different topologies, and we show the mean and 95% confidence interval for the evaluation metrics. The GeoCHT compression of local coverage area descriptions is not used in any of the scenarios.

Average query delivery delay (AQD) of both protocols is first recorded as the end-to-end delay (the time between query generation and reception of the query in the target region). However, AQD results show that there is a large difference between the query delivery delays of GEAR and GeoCHT. This is mainly due to the current *NS2* implementation of GEAR as an extension to existing directed diffusion algorithm [79]. Because an interest message in GEAR passes through a series of filters and agents, a query spends a lot of time between agents resulting in high end-to-end delay. As a result, GEAR delay data is not directly comparable with our protocol's delay data. Therefore, we record the *hop-based transmission delays of* *a message* (i.e. sum of the times between sending the message by a node and receiving the message by the next node for every hop of the routing path) for GEAR, which excludes the time spent in the routing agents. Only in Figure 4.6(a), we show also the end-to-end delay of GEAR to present its difference with hop-based delay.

4.6.3 Static networks

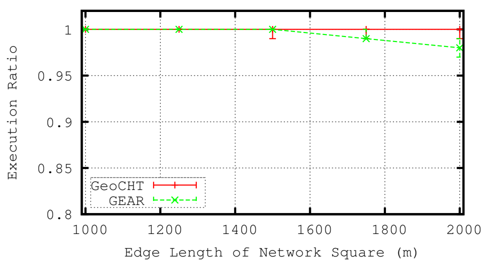
In this section, we evaluate the proposed geographical routing GeoCHT in static networks without any disturbing factors (scenarios 1 to 3). We introduce errors in the position estimates of nodes in scenario 4. Simulations of all scenarios show that the FER of queries in static networks is zero for both GEAR and our protocol GeoCHT, meaning that none of the nodes outside the area of interest executes the query. Therefore, FER results are not shown in the graphs.

Scenario 1: Effects of network density

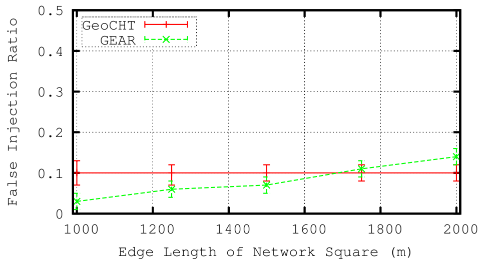
Scenario 1 evaluates the effect of varying network density by varying edge length of the (square) deployment area from $1000m$ to $2000m$. The number of nodes and sinks is fixed, 100 nodes and 3 sinks. Figure 4.5 shows the routing accuracy performance of this scenario.

In Figure 4.5(a), we compare the ER of GEAR and GeoCHT for varying network density. Note that an execution ratio of 1 means that every node inside the target region has received and executed the corresponding query packet. GEAR protocol, which has a higher redundancy than GeoCHT, has a lower execution ratio in sparse networks. This is mainly due to the unicast nature of the GEAR protocol. Its Greedy forwarding may not be able to find a path between a sink and the target region when the network is sparse. On the other hand, even in sparse networks, in GeoCHT, the sinks can form their coverage areas including all sensors in the network. Therefore, nodes in the target region are accessible at least from one of the sinks in the network.

Some false injections exist even in static networks. Figure 4.5(b) compares the FIR of the two protocols. FIR of GeoCHT is constant because the protocol eliminates the unnecessary query injections from sinks both in dense and sparse networks. GeoCHT has always a small number of false injections due to the fact that the convex hull of a sink can overlap with the region of interest although no node from its partition is present in the target area. This effect is due to our choice of describing the coverage area with a convex hull. In dense networks, FIR

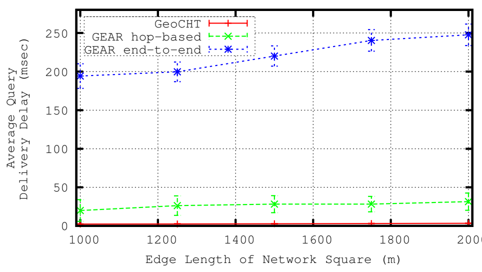


(a) Execution Ratio

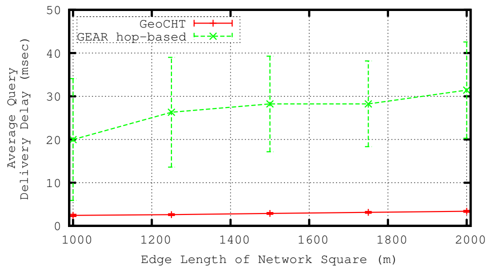


(b) False Injection Ratio

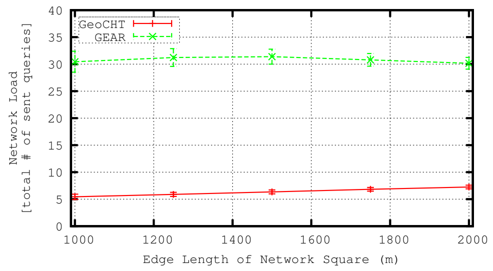
Figure 4.5: *Scenario 1 – Routing accuracy performance of varying network density scenario*



(a) Average Query Delivery Delay (msec)



(b) Average Query Delivery Delay (msec)



(c) Network Load (number of sent packets)

Figure 4.6: *Scenario 1 – Networking performance of varying network density scenario*

of GEAR is less than FIR of GeoCHT. However, when the network is getting more sparse, FIR of GEAR increases. This is because all the sinks send the query packet, and the Greedy forwarding uses it for routing to the region of interest fails more often in sparse networks.

Figure 4.6 shows the networking performance of *Scenario 1*. Figure 4.6(a) shows the delay of GeoCHT, end-to-end delay of GEAR and hop-based delay of GEAR. Due to many agents called in GEAR (as explained in Section 4.6.2) the difference between GEAR end-to-end delay and hop-based delay is very big.

Figure 4.6(b) presents the hop-based delay of GEAR and end-to-end delay of GeoCHT. The end-to-end delay of GeoCHT is around 2 msec for a network of 1000x1000m^2 and 4 msec for a network of 2000x2000m^2. The delay of GeoCHT increases slightly with the increase of the network area. GeoCHT delay is smaller than GEAR delay because GeoCHT discards the queries of sinks which coverage area does not intersect with the region of interest. Only the sinks with overlapping coverage area with the target region inject their queries. These sinks are often closer to the region of interest than the other sinks. This keeps the average query delay low. In GEAR all the sinks inject the query, therefore the average query delivery delay gets higher. Another reason of this difference in delays of GEAR and GeoCHT is their different forwarding mechanisms. GEAR uses unicasting to reach the next hop, and GeoCHT uses a special restricted broadcasting, which is quicker in the delivery of the message to the next node. Also, in the GEAR's delay graph, the 95% confidence range is very large, which shows a different response for different topology configurations.

Figure 4.6(c) shows the network load of both protocols. GEAR generates more network load than GeoCHT, since all of the sinks in GEAR try to reach the target region and inject a query packet into the network. The multiple forwarding paths also result in higher network load in GEAR. On the other hand, GeoCHT checks the coverage areas of sinks and a sink sends a query packet only if its coverage area overlaps with the region of interest. Therefore, GeoCHT sends less query packets towards the target region. We can also say that GeoCHT is more energy efficient that GEAR because it sends less query packets, while having a very high query execution ratio.

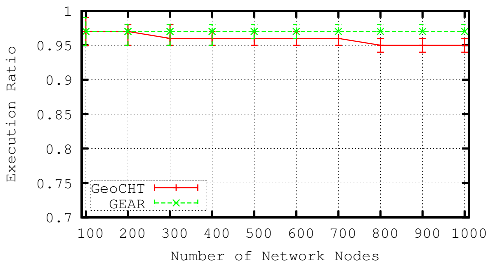
Scenario 2: Effect of varying network size

This scenario evaluates the effect of varying network size (from 100 to 1000 nodes), while the mean node density and the mean sink load are kept constant. The average node density is set to 6 neighbors per node and the average sink load —number of nodes connected to a sink– is set to 33 nodes per sink. In this scenario, the network area is increased with the number of nodes. For 100 nodes, the network size is 1800x1800m^2; for 1000 nodes, it is 5700x5700m^2. To keep the sink load constant we increase the number of sinks: 3 sinks in the network with 100 nodes, 30 sinks in the network with 1000 nodes. Figure 4.7 shows the routing accuracy performance of this scenario.

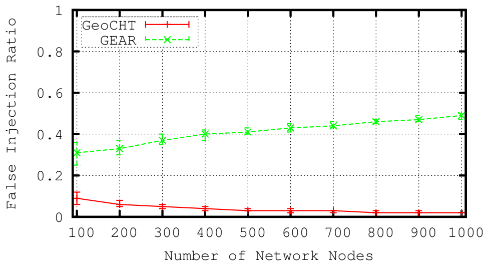
In Figure 4.7(a) we compare ER of GEAR and GeoCHT for varying number of network nodes. The ER of GEAR slightly outperforms the ER of GeoCHT when the number of network nodes increases. GeoCHT approach is less redundant than GEAR since in GEAR all sinks send the query towards the target region. Therefore, queries in GEAR reach to the target region via multiple paths, which results in higher ER.

Looking at the FIR of the two protocols, Figure 4.7(b), we see that FIR of GEAR is getting higher when the number of network nodes increases. Since we also increase the number of sink nodes, more sinks inject the query packets in GEAR. Therefore, the network with more sinks has a higher probability of query packets not reaching to the area of interest. On the other hand, the FIR of GeoCHT decreases as the number of sink nodes increases because more sinks mean more coverage areas that partition the network area, and the probability of the target region overlapping a coverage area decreases. In GeoCHT, sinks can decide to discard the query without inserting it in the WSN. Thus, GeoCHT can eliminate more unnecessary query injections in networks with more sinks and more nodes. FIR results show that the ratio of unnecessary injections of queries is very low in GeoCHT while GEAR has many unnecessary query insertions, which consume more energy and shorten the lifetime of the WSN.

Figure 4.8 shows the networking performance for varying number of nodes. As it can be seen in Figure 4.8(a), the query delivery delay of GEAR is increasing excessively with the number of network nodes, while the delay of GeoCHT is stable. The reasons are the same as for the previous scenario. GeoCHT only injects the queries of sinks which coverage areas overlap with the region of interest. These sinks are usually closer to the region of interest, therefore the paths from sinks to the target region are shorter. GEAR sends the queries from all sinks. When the network size increases, it results in more hops between the farthest sink and the target region, therefore longer delays for GEAR. Figure 4.8(b) presents the network load of the protocols. We notice that in both protocols, the network load increases with the increase of the number of nodes. However, the increase in network load of GEAR is very large since more sinks generate more query packets and all these query packets try to

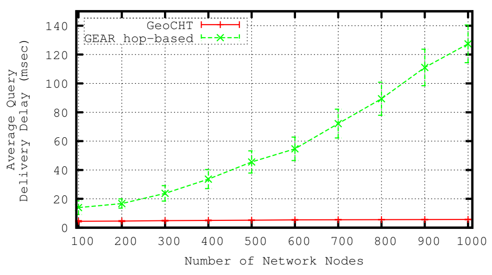


(a) Execution Ratio

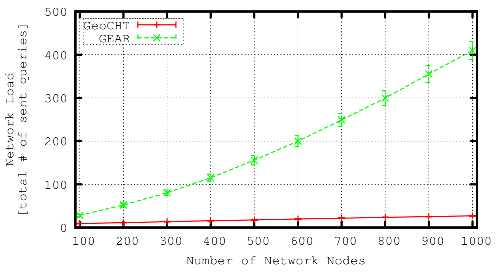


(b) False Injection Ratio

Figure 4.7: *Scenario 2 – Routing accuracy performance of varying network size scenario*



(a) Average Query Delivery Delay (msec)



(b) Network Load (number of sent packets)

Figure 4.8: *Scenario 2 – Networking performance of varying network size scenario*

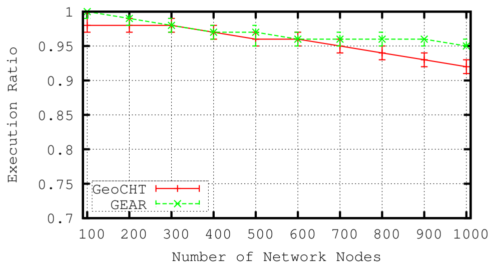
reach the area of interest. GeoCHT, on the other hand, eliminates most of the unnecessary query injections, thus has a very small increase in the network load as the number of nodes and sinks increase. Summarizing results of Figure 4.7 and Figure 4.8, we can conclude that GeoCHT has a large energy saving even in big networks. Although GEAR uses multiple paths to forward queries towards the area of interest, the query execution ratio of GeoCHT is very close to the ER of GEAR.

Scenario 3: Effects of sink load

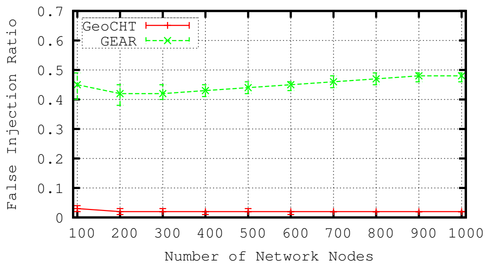
Scenario 3 evaluates the effect of the number of nodes associated per sink in the network. This measures how the performance changes with the sink load, which determines at the same time the average path length in the network. The sink load is varied between an average of 11 nodes/sink and 110 nodes/sink. The node density is kept on average at 6 neighbor nodes and 9 sinks are used in the network setup. Both the deployment area and the number of nodes in the network vary in order to tune the sink load.

Figure 4.9 shows the routing accuracy performance of scenario 3 to see the effect of varying sink load. As shown in Figure 4.9(a), ER of GEAR and GeoCHT decrease when the load (the number of nodes) per sink increases. ER of GEAR and ER of GeoCHT are very close to each other, but GEAR performs again better in this metric.

False injection ratios of both protocols remain more or less constant as the sink load

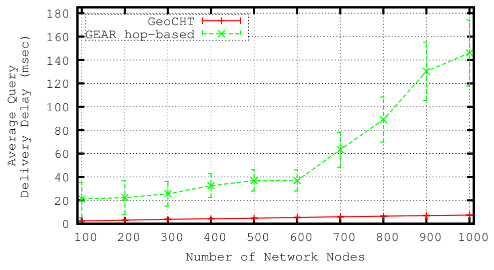


(a) Execution Ratio

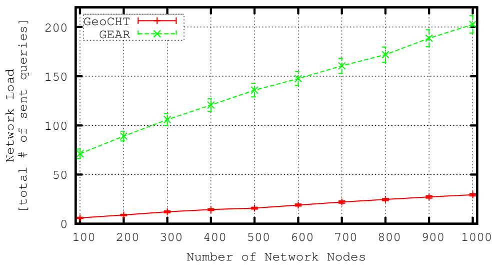


(b) False Injection Ratio

Figure 4.9: *Scenario 3 – Routing accuracy performance of varying sink load scenario*



(a) Average query delivery delay (msec)



(b) Network Load (number of sent packets)

Figure 4.10: *Scenario 3 – Networking performance of varying sink load scenario*

increases, as shown in Figure 4.9(b). This is because the number of sinks is constant. GEAR has a higher FIR than GeoCHT. This is mainly due to the fact that GeoCHT eliminates unnecessary query injections by checking if the coverage area of a sink intersects with the target region. On the other hand, GEAR injects the queries from all sinks even though they may be far from the target region. The transmission of a query packet injected by the far away sinks may fail in the network due to the failure of pure Greedy forwarding at dead ends.

Figure 4.10(a) presents how the query delivery delays of both protocols are affected by the varying sink load. When we increase the sink load, the delay of GeoCHT slightly increases but the delay of GEAR increases very much. Having a fixed number of sinks but increasing number of network nodes in a constant node density network, results in longer paths between sinks and target region. This effect is seen more clearly in the delay graph of GEAR because all sinks inject their queries into the network. Figure 4.10(b) shows the network load of the protocols when we increase the sink load. For the same reasons explained in previous scenarios, the network load of GEAR is very high because all the sinks inject the query in the network. In GeoCHT only few sinks inject the query into the WSN, usually those that are close the region of interest. The longer paths are another reason for network load. This is more visible in GEAR because all the sinks send a query towards the target region, even those that are far away.

4.6.4 Sensitivity to position estimate error

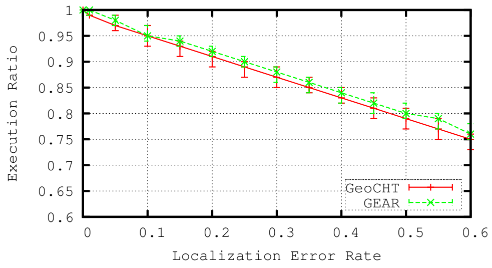
In this section we simulate positioning errors in static networks. In GeoCHT and GEAR is assumed that nodes are able to estimate their positions. However, most localization schemes introduce errors. As a consequence, in GeoCHT the service areas do not match the reality exactly and the execute decisions get less accurate, resulting in sub-optimal performance.

To investigate the sensitivity of both protocols to position estimate errors, we examine the impact of random location errors for every node in the network. The error in location was generated uniformly in the range [−error-rate * transmission-range, error-rate * transmission-range]. For example, if the error rate is 0.1 and the transmission range is 250m, the randomized error is uniformly distributed in [-25, 25]. The following simulations are performed in a 1000x1000m^2 network having 100 nodes and 3 sinks. The error rate varies between 0 (i.e. no positioning errors) to 0.6, thus the randomized error is between $-150m$ and $150m$.

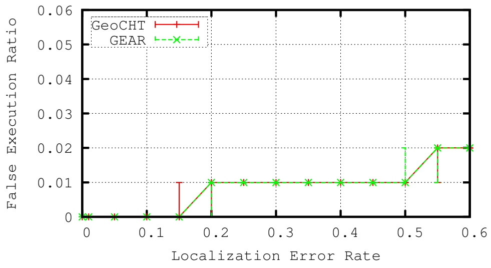
Figure 4.11 shows the routing accuracy performance of varying position estimate error rates. The results in Figure 4.11(a) show that ER decreases with increasing position estimation errors. However, with relatively big location errors, both GeoCHT and GEAR protocols still achieve satisfying performance. For example, when we introduce random errors in the range [-25, 25] for each node's x- and y-coordinates (i.e. error rate of 0.1), the simulation results show very small performance degradation: the ER is still 95% for both protocols. When we introduce a randomized error between [-150, 150] for each node's x- and y- coordinates (i.e. error rate of 0.6), the simulation results show around 25% performance degradation in ER. The sensitivity of both protocols to position estimate errors is very similar and ER of GEAR and GeoCHT are very close to each other.

Since we introduce localization errors in the network, some false query executions exist in the network as shown in Figure 4.11(b). False execution ratios of the protocols are the same for a given position estimate error rate. For an error rate of 0.15, i.e. around 37.5m error, FER of GEAR and GeoCHT is still zero. If we introduce an error rate of 0.2, 1% of the nodes outside the region of interest are executing the queries on average. FIR results shown in Figure 4.11(c) are not much affected from the increase of the positioning error. FIR of GeoCHT is slightly higher than FIR of GEAR. However, when we increase the localization error rate, the FIRs of both protocols are more or less the same. Coverage region based approach of GeoCHT allows better hiding of inaccuracy of node positions for higher localization errors. On the other hand, using individual inaccurate positions for geographical routing in GEAR results in more failures of query forwarding for higher error rates.

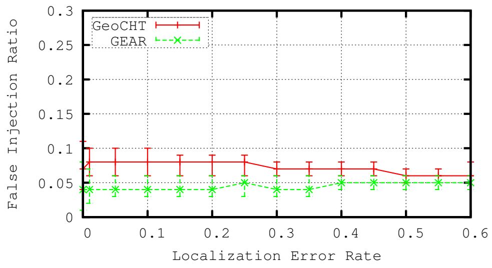
Figure 4.12 shows the networking performance of varying position estimate error. In Figure 4.12(a), query delivery delays of GEAR and GeoCHT are shown. The query delivery delay of GEAR increases progressively when the localization error rate increases. On the other hand, GeoCHT has a constant delay with increasing localization error rate. GeoCHT is based on convex hulls of sinks' coverage areas and even if we have localization errors in nodes, the resulting convex hulls are the approximation of the correct convex hulls. Since we have a hierarchical structure in the definition of coverage areas in GeoCHT, a node has an 'overview' about the region between its position and the area of interest. This 'overview' is affected by localization errors, but is still a good approximation of the correct coverage area. Therefore, the path between a sink and the area of interest is also an approximation of the correct path between the sink and the area of interest. As a results, the delay of GeoCHT is not affected by increasing localization errors.



(a) Execution Ratio

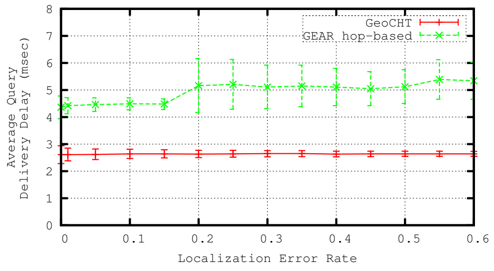


(b) False Execution Ratio

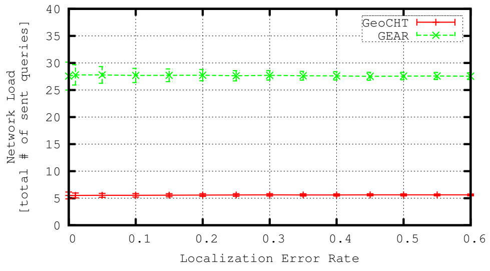


(c) False Injection Ratio

Figure 4.11: *Scenario 4 – Routing accuracy performance of varying position estimate error rates*



(a) Average query delivery delay (msec)



(b) Network Load (number of sent packets)

Figure 4.12: *Scenario 4 – Networking performance of varying position estimate error rates scenario*

However, routing strategies such as GEAR, which carry the packet geographically closer to the destination in each hop, can result in different paths with different localization errors. Since a node in GEAR only has a local knowledge about its surrounding nodes (i.e. one hop neighbors' positions) and it forwards the message based on this local knowledge, the path between sink and the target region can be longer when nodes have position estimate errors. In Figure 4.12(a), we observe higher delays with higher localization errors in GEAR, and we expect that to be caused by longer paths between the sinks and the nodes in the target region.

The network loads of both protocols are constant for varying error rates. Even if we have position estimate errors in the network, both GEAR and GeoCHT try to forward queries towards the area of interest. Therefore, the network load is constant when we increase the error rate. The number of packets the protocol sends when there is no positioning error in the network is the same with the number of packets sent when there are positioning errors. Although network load of GEAR and GeoCHT do not change with the increase of error rate, the ER of both protocols is affected, because the protocols cannot deliver the query packets to the correct locations due to positioning errors.

4.6.5 Mobile networks

The effects of mobility on both protocols are evaluated in scenarios 5 and 6. The mobile network simulations are performed in a 2500x2500m^2 deployment area with 200 sensor nodes and 6 sinks. Simulations of both mobile scenarios show that the False Execution Ratio of queries in mobile networks is zero for both GEAR and GeoCHT protocols, therefore not shown in the graphs. This is due to the fact that when a mobile node that was in the target region before starting to move, receives a query, it first checks its current position and if it is not inside the area of interest anymore, it does not execute the query.

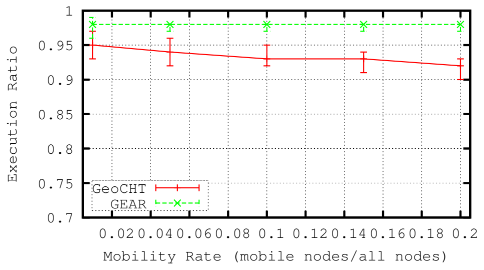
Scenario 5: Effect of the number of mobile nodes

This scenario evaluates the effects of the different number of mobile nodes in a network: 1% to 20% of the network nodes are moving. The average speed of the mobile nodes is $5m/s$. Figure 4.13 shows the routing accuracy performance of this scenario.

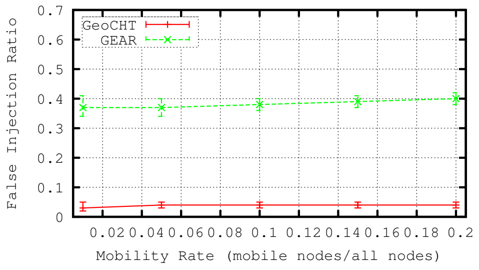
Figure 4.13(a) shows the ER results for GEAR and GeoCHT. GEAR slightly outperforms GeoCHT when we increase the mobility rate. This is mainly due to the fact that we used tree-based shortest path routing to connect nodes to sinks in GeoCHT simulations. As tree-based approaches require frequent reconfigurations in mobile sensor networks, they may have worse performance in mobile sensor networks.

Figure 4.13(b) presents the FIR results of GEAR and GeoCHT. As we already observe in the previous simulations, the FIR of GEAR is higher than the FIR of GeoCHT. When we increase the mobility rate in the network, the FIR of GEAR slightly increases. GeoCHT also has a slight increase in FIR when the mobility rate is increased. This means that although convex hulls of sensors may be affected by mobile nodes, the coverage areas of sinks are not much affected. Only the movement of nodes that are close to the boundary of a sink's convex hull may change the coverage area of this sink when they pass this boundary.

Figure 4.14 shows the networking performance of this scenario. In Figure 4.14(a) we show the query delivery delays of both protocols. The delays are not affected much by the increasing number of mobile nodes in the network. The delay is still much higher in GEAR than in GeoCHT. The network loads of GEAR and GeoCHT are shown in Figure 4.14(b). As we already see in the previous simulations, the network load of GEAR is higher than the network load of GeoCHT, also in this mobile scenario. The network load of GEAR is slightly decreasing with the increase of the number of mobile nodes. This is to be seen together with the ER of GEAR. The probability of a neighborhood change is getting higher with the increase of mobility rate. A query packet can get stuck at a node that does not have anymore a neighbor closer to the region of interest. Therefore, the number of sent query packets decreases in GEAR.

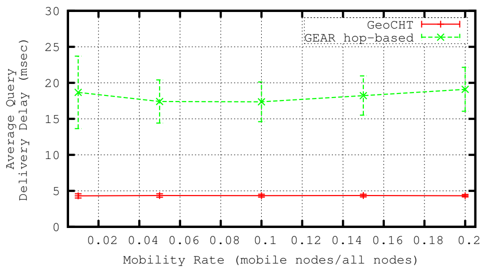


(a) Execution Ratio

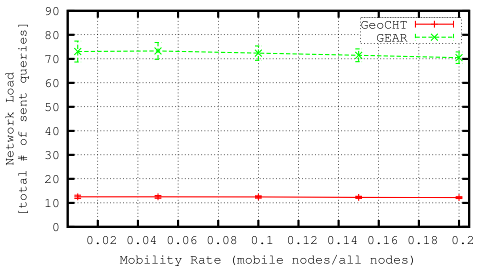


(b) False Injection Ratio

Figure 4.13: *Scenario 5 – Routing accuracy performance of varying mobility rate scenario*



(a) Average Query Delivery Delay (msec)



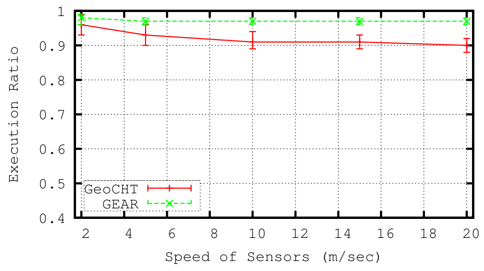
(b) Network Load (number of sent packets)

Figure 4.14: *Scenario 5 – Networking performance of varying mobility rate scenario*

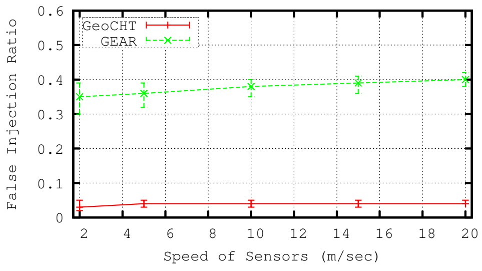
Scenario 6: Effect of speed of mobile nodes

The speed of mobile nodes is varied in this scenario: 5% of the network nodes are moving with a speed varying between $2m/s$ and $20m/s$. Figure 4.15 shows the routing accuracy performance. The results in Figure 4.15(a) and Figure 4.15(b) show the same behavior as the simulation results of scenario 5. Higher mobility speed requires more frequent reconfigurations in mobile sensor networks in GeoCHT. Therefore, ER of GeoCHT is less than ER of GEAR due to the tree-based structure used by GeoCHT, but the difference is very small. FIR results of GEAR and GeoCHT in scenario 6 are also very similar to the results in scenario 5.

Figure 4.16(a) presents the query delivery delays of both protocols when we increase the speed of mobile sensors. The query delay of both protocols is affected very little by the increase on nodes speed. GeoCHT is outperforming GEAR considerably. Figure 4.16(b) shows the effect of nodes speed on the network load. GeoCHT network load is unaffected by the change in mobile nodes speed. It is again much lower than the network load of GEAR.

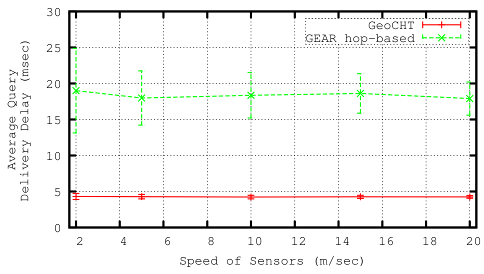


(a) Execution Ratio

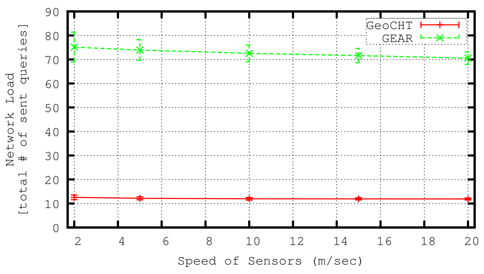


(b) False Injection Ratio

Figure 4.15: *Scenario 6 – Routing accuracy performance of varying mobility speed scenario*



(a) Average Query Delivery Delay (msec)



(b) Network Load (number of sent packets)

Figure 4.16: *Scenario 6 – Networking performance of varying mobility speed scenario*

4.7 Local Updates to Handle Mobility of Sensors

Global updates periodically rebuild all routing trees rooted at the sinks and the local convex hulls of all the subtrees. This approach updates even the tree branches which are not affected by the mobile nodes. In this section, we introduce *local updates* initiated by the mobile sensors to update only the branches affected by mobility of nodes.

When a mobile node (MN) moves from one location to another, it may change its neighborhood, including its parent and child nodes. As result of mobility, the child nodes of the MN may need to re-associate themselves to another parent node and/or the MN may need to re-associate itself to a new parent node. Figure 4.17 shows which nodes and local convex hulls are affected by the mobility of a node. MN and one of the child nodes (i.e. C_2) have to associate themselves with new parent nodes (i.e. P_{New} and PC_{new}) after the movement of MN. Also, the convex hulls of new parent nodes (i.e. P_{New} and PC_{new}), and old parent nodes (i.e. P_{old} and MN) change due to mobility. In the following we explain how to detect these changes by the neighbors of MN and how to update accordingly the convex hulls of the affected nodes.

As shown in Figure 4.18, the neighbor list of a MN consists of the child nodes (i.e. Node C_1 and C_2) and a parent node (i.e. Node P) of MN and the other close-by nodes (e.g., Node N). In our local updating approach, to inform its neighbors about its movement, MN periodically broadcasts *beacon* packets called MN_BEACON. The frequency of MN beaconing depends on the transmission range, speed of MN, and a constant, k (i.e. *beaconing constant*) defined by the network. The following equation shows how the frequency of MN beacon packets is calculated:

$$MN\_BEACON\_INTERVAL = \frac{R_{TX}}{V_{MN}} \times k \tag{4.6}$$

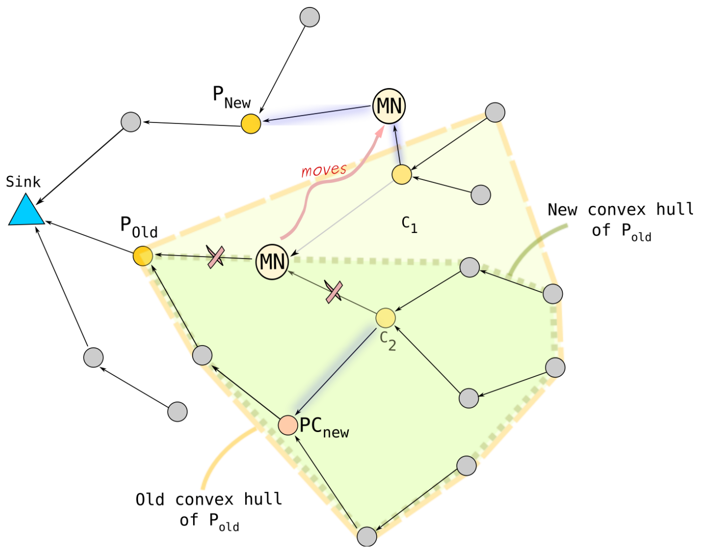


Figure 4.17: *Effects of movement of MN on the routing tree and convex hulls*

where $0 < k \leq 1$. V_{MN} and R_{TX} denote the maximum mobile node speed (m/s) and the maximum radio range (m). $\frac{R_{TX}}{V_{MN}}$ is the time to leave the transmission range. The constant k helps to determine how often a MN_BEACON packet should be sent. It can be designated based on the network characteristics such as node density. We assume that the radio range shown in Figure 4.18 is the same for both sinks and sensor nodes. It is important to note that MN_BEACON_INTERVAL is the local update rate since it is determined by the speed of MN. Figure 4.18 shows the content of a MN beacon packet. A MN_BEACON packet contains the ID of MN, frequency of the beaconing, and parent ID of MN.

Messaging during the local updates is illustrated in Figure 4.19. In the figure, MN broadcasts a MN_BEACON packet at time t_0. The receivers of a beacon packet first check who sent the beacon. The receiver node can be (i) the parent node P of MN, or (ii) a regular neighboring node N of MN, or (iii) a child node C of MN. Every node that receives a MN_BEACON packet sends an acknowledgement (i.e. ACK) message back to MN initiating the beaconing as shown in Figure 4.18. However, the content of the acknowledgement message is determined by the relationship between MN and the receiver node:

- When a regular neighboring node N or the parent node P of MN receives a MN_BEACON packet, the receiver node sends an ACK_BEACON packet back to MN. Figure 4.19 shows that the parent node P and the neighbor node N send ACK_BEACON packets back to MN at times t_1 and t_2, respectively. An ACK_BEACON packet includes the ID of the node, the sink ID which this node is connected to, and the number of hops to the sink as shown in Figure 4.18.
- When one of the child nodes of MN receives a beacon from MN, the child node sends an

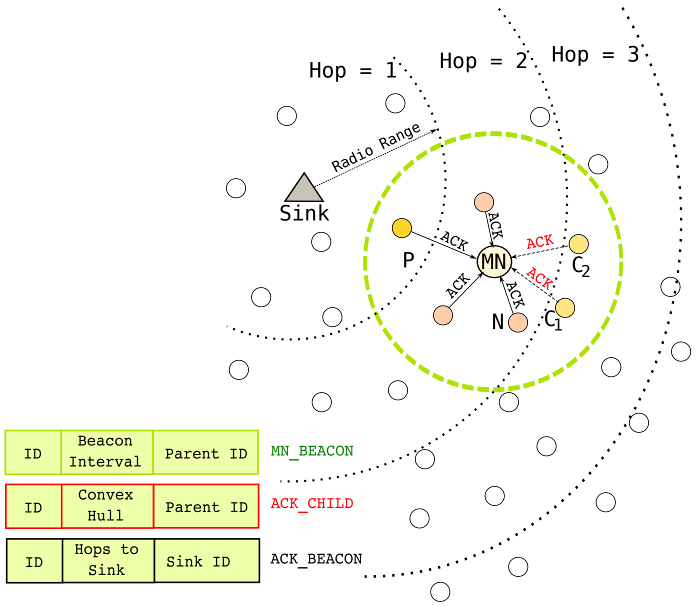


Figure 4.18: *Beaconing to handle mobility of MN in the routing tree*

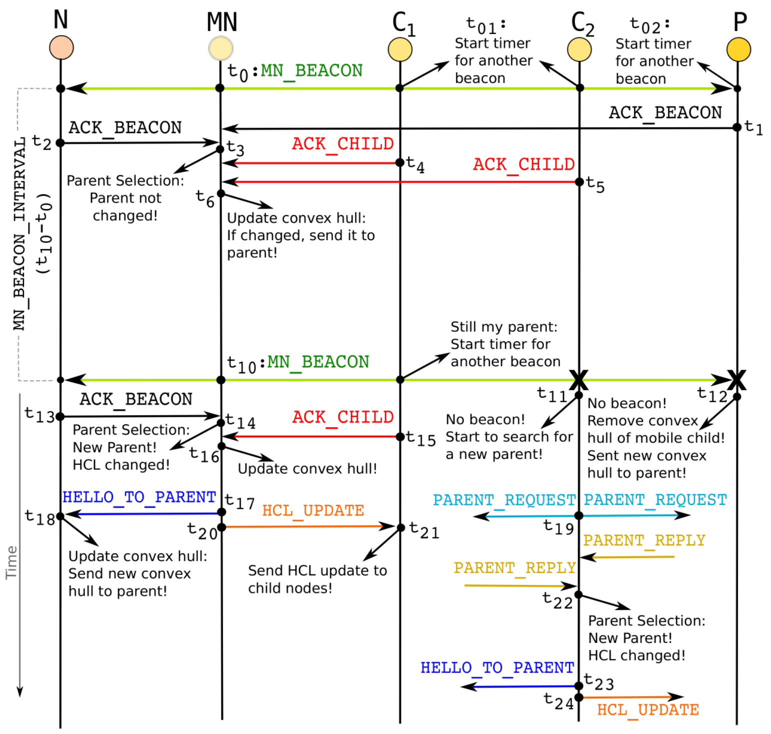

N
MN
C1
C2
P
t01: Start timer for another beacon
t02: Start timer for another beacon
t0:MN_BEACON
ACK_BEACON
t1
ACK_BEACON
t2
t3
ACK_CHILD
t4
ACK_CHILD
t5
Parent Selection: Parent not changed!
t6
Update convex hull: If changed, send it to parent!
MN_BEACON_INTERVAL (t10−t0)
Still my parent: Start timer for another beacon
t10:MN_BEACON
t11
t12
No beacon! Start to search for a new parent!
No beacon! Remove convex hull of mobile child! Sent new convex hull to parent!
ACK_BEACON
t13
t14
ACK_CHILD
t15
Parent Selection: New Parent! HCL changed!
t16
Update convex hull!
HELLO_TO_PARENT
t17
HCL_UPDATE
t18
t20
t21
PARENT_REQUEST
PARENT_REQUEST
t19
PARENT_REPLY
PARENT_REPLY
Update convex hull: Send new convex hull to parent!
Send HCL update to child nodes!
t22
Parent Selection: New Parent! HCL changed!
HELLO_TO_PARENT
t23
t24
HCL_UPDATE
Time

Figure 4.19: *Messaging during local updates*

ACK_CHILD packet back to MN. Figure 4.19 shows that the child nodes C_1 and C_2 send ACK_CHILD packets back to MN at times t_4 and t_5, respectively. An ACK_CHILD message includes the convex hull (i.e. coordinates of convex hull) of the child node as shown in Figure 4.18.

It is also important to point out that when the parent node or a child node of MN receives a beacon, it starts to wait for another beacon from MN (at times t_{01} and t_{02} in Figure 4.19). After the beacon interval passes, MN sends another MN_BEACON packet at time t_{10} as shown in Figure 4.19. If the child node of MN does not get another beacon (since MN moves out of communication range of the child node) in the time period (beacon interval) specified in the MN_BEACON packet, it starts to *search for another parent* at time t_{11} (details in Section 4.7.5). If the parent node of MN does not receive another beacon within the beacon interval (since MN moves out of communication range of the parent node), it *removes the convex hull of MN form its local convex hull* at time t_{12} (details in Section 4.7.4).

After receiving ACK packets from its neighbors, MN executes the following actions if needed: (1) Updating the convex hull and propagating the changes up on the tree if MN does not get an ACK from a child node, (2) New parent selection if MN does not get an ACK from the parent node, and (3) Propagation of the convex hull and the new hop count level after MN selects a new parent.

4.7.1 Updating the convex hull of MN and propagation of changes

If a child node of a MN is not in the communication range of MN anymore, MN does not receive ACK packets from this child node. Therefore, the convex hull of this child node should be removed for the convex hull of MN. For a MN to maintain its convex hull, it calculates its new convex hull after it receives new ACK_CHILD messages from its child nodes which are still connected to MN (at times t_6 and t_{16} in Figure 4.19). There are two options for updating MN's convex hull: (i) MN can use a timeout mechanism by utilizing ACK_CHILD packets; if there is no ACK_CHILD for a given interval from a specific child, MN removes the convex hull of this child (a similar approach explained in Section 4.3.2), or (ii) it can wait for a predefined time interval and then calculates its convex hull from the received ACK_CHILD packets. If MN's convex hull is changed[‡], MN sends a convex hull update message (i.e. HELLO_TO_PARENT) to its parent to inform the parent about its new convex hull as shown in Figure 4.19 at time t_{17}. Any node receiving a convex hull update message first checks if the aggregated (local) convex hull is changed. If the node's convex hull is also changed, then it sends also an update message (at time t_{18} in the figure) to its parent until the update message reaches to the root of the tree, the sink node.

4.7.2 New parent node selection of MN

If MN receives an ACK_BEACON packet from its parent, MN concludes that it is still in the communication range of its parent. Although MN is still in the communication range of the current parent node, it may change its *hop count level* (HCL) due to its mobility, e.g. comes closer to the sink. However, as long as the current parent of MN is in the communication range, MN keeps the same parent node although there exist other parent candidates having shorter distances to the sink. With this approach, it is possible to avoid sending of extra HCL update messages (i.e. HCL_UPDATE) to the child nodes of MN after changing the parent and HCL of MN. Also, MN still remains connected to the tree.

If MN does not receive an ACK_BEACON packet from its parent, MN concludes that it is not in the communication range of its parent anymore. Therefore, MN checks the HCLs (i.e. hops to sink entries in ACK packets) of other ACK_BEACON packets sent by other neighbor nodes. When MN receives an ACK_BEACON, it records the sender node as a candidate for the parent node. After the predefined time interval expires, MN checks its candidate parents list. Since its current parent is not in the list, it chooses the node which has the smallest HCL to the sink from the candidate parents list as the new parent node. In Figure 4.19 at time t_{14}, MN chooses node N as the new parent.

4.7.3 Convex hull and HCL propagation after MN changes its parent

If the hop count level (HCL) of MN changes after the new parent selection, MN sends its new HCL (in HCL_UPDATE message at time t_{20} in Figure 4.19) to its current child nodes. Therefore, the child nodes can also update their HCLs to the sink. This HCL update message is propagated (at time t_{21} in the figure) until it reaches to the leaf nodes of the branch. Also, MN sends a HELLO_TO_PARENT message including it convex hull to its new parent. The parent node receiving a HELLO_TO_PARENT message recalculates its convex hull. The convex hull changes are propagated until the changes reach to the sink.

[‡]To check if its convex hull is changed, MN stores the previous convex hull. After comparison, it deletes the old convex hull.

After receiving the first beacon packets from MN, child nodes and the parent node of MN start to wait for another MN beacon packet for a beacon interval. In the following, we explain the actions executed by the parent and child nodes of MN when there is no other beacon from MN.

4.7.4 Removing MN's convex hull from its previous parent node

If the parent node of a MN does not receive another beacon packet from its mobile child after the beacon interval defined in the previous MN_BEACON packet, it concludes that it is not in the communication range of its mobile child anymore. Therefore, the convex hull of the mobile child should be removed form the convex hull of the previous parent of MN as illustrated in Figure 4.19 at time t_{12}. The timeout mechanism for removing coordinates from local coverage areas is also executed here to remove MN's convex hull from the convex hull of its previous parent. It is important to point out that the timeout interval should not be shorter than the beaconing interval (i.e. MN_BEACON_INTERVAL) to ensure not to remove MN's convex hull before it is reinforced.

4.7.5 Parent invalidation for child nodes of MN

If a child node of MN does not receive another beacon packet from its mobile parent after the beacon interval defined in the previous MN_BEACON packet, it concludes that it is not in the communication range of its parent anymore. Therefore, it searches for another parent. For this purpose, it broadcasts a PARENT_REQUEST message. The child node C_2 does not receive another beacon from MN and at time t_{19} it starts to search for another parent by sending a PARENT_REQUEST message in Figure 4.19. All nodes receiving a PARENT_REQUEST message reply a PARENT_REPLY message back to the initiator of the parent request message. After the node receives PARENT_REPLY messages, it chooses the node which has the smallest HCL to the sink (at time t_{22} in the figure). Then, the child node sends a HELLO_TO_PARENT message to its new parent to inform about its convex hull (at time t_{23} in the figure). The new parent receiving a HELLO_TO_PARENT combines its convex hull with its new child node's convex hull and propagates its new convex hull to upper nodes in the tree if the aggregated convex hull is changed. After the new parent selection, if the current HCL of the child node is different than its previous HCL, it should also propagate its new HCL to its child nodes (at time t_{24} in the figure) and the whole branch down in the tree.

4.8 Performance Evaluations of Local Updating

In order to evaluate the performance of global and local updates described above, we used the open source network simulator NS-2 [15] version 2.33. We have added a new routing agent (i.e. geocast) into NS-2 over the currently implemented network stack.

In the following, we provide first the descriptions of the scenarios and the metrics for evaluating the performance of the proposed mobility handling methods and for evaluating the routing accuracy and networking performance of the convex hull based geocasting. After that, we present the scenarios characteristics of mobility handling simulations. Finally, we analyze the simulation results we obtained.

4.8.1 Evaluation Scenarios and Metrics

The proposed mobility handling method (i.e. local updates) for our geographical routing protocol is evaluated in terms of routing accuracy i.e. how well the proposed mechanism delivers messages to the region of interest defined in a query, and in terms of communication overhead. The performance of two methods (i.e. global updates and local updates) for handling mobility in geocasting has been evaluated by varying the number of nodes, number of mobile nodes and speed of the mobile nodes with the following scenarios:

- *Scenario 1* – We vary the number of nodes from 60 to 360, where 10% of the nodes are moving with a speed of around $10m/s$.
- *Scenario 2* – We vary the number of mobile nodes from 10 to 50 in a network having 200 nodes. The speed of the mobile nodes is around $10m/s$.
- *Scenario 3* – We vary the speed of mobile nodes from 1 to $17m/s$ in a network having 200 nodes with 20 mobile nodes.

We have measured two metrics to evaluate the mobility handling mechanism for routing of queries towards a specific region in these multi-sink WSN scenarios:

- *Execution ratio (ER)* – ER is explained in Section 4.6.1.
- *Communication Overhead (CO)* – The total number of packets that are sent to construct and maintain the structure. It includes hello packets sent from sinks, beacon packets sent from mobile nodes and convex hull definition update messages. If more messages need to be transmitted to keep the geocast structure up-to-date, the energy consumption in the WSN is likely to increase. We measure this effect with the overall network load.

False injection ratio (FIR), False execution ratio (FER), Average query delay (AQD), Network load (NL) are used to evaluate the performance of convex hull based geocasting in Section 4.6. These metrics are not much affected by mobility. Since our aim is to evaluate the performance of discussed mobility handling techniques in a tree-based network, in this section we are mostly interested in ER and CO which are directly related with mobility handling.

4.8.2 Scenario Characteristics for Mobility Handling Simulations

In the simulations, the sensor nodes are uniformly deployed in a rectangular deployment area of $1800x1800m^2$ where we have 3 sinks. Every sensor node has a transmission range of $250m$. We use Random Waypoint movement model to simulate sensor node mobility. The speed of mobile nodes varies from low mobility (i.e. 4-5 km/h for walking humans) to high mobility (e.g., 40-60 km/h for UAVs[§]) scenarios. A simulation run lasts 100 simulation seconds. For a given simulation run, results are averaged over all the queries sent during the simulation. The averaged values are calculated over 10 different deployments with a fixed number of sensor nodes. For each deployment, we randomly select 5 different center points for disjoint target regions which are circular areas. Assuming to divide the deployment

[§]The typical UAV speed is taken from the specification document of EU project AWARE (IST-2006-33579, http://www.aware-project.net) and were used in the field experiments of AWARE in May 2009 in Seville, Spain.

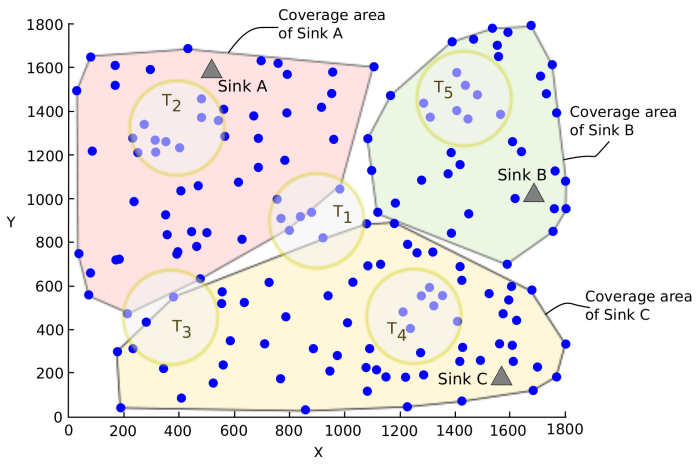


Figure 4.20: *An example topology with coverage areas of sinks and target regions*

Table 4.3: *Simulation Parameters*

Simulation time	$100s$
Simulation area	$1800x1800m^2$
MAC protocol	IEEE 802.11 DCF
Transmission range	$250m$
Number of sinks	3
Number of sensor nodes	60 to 360
Number of mobile nodes	10 to 50
Mobile node speed	1 to 17 m/s (4 to 60 km/h)
Mobility model	Random Waypoint
Global update frequency, f	$3s$ to $25s$
Mobile node beacon constant, k	0.05, 0.10, 0.20
Target region	Circle, radius $250m$
Target region selection	5 random center points
Query sending frequency	$10s$ for local update simulations $100ms$ before global updates

area into 4 sub-rectangles from the center, one of the target region is at the middle of the network and the other four are in one of the sub-rectangles. Figure 4.20 illustrates an example simulation topology with convex hulls of sinks and target regions. The target regions are created in a way that they cover all the directions in the deployment area.The simulation parameters are given in Table 4.3.

We compare the performance of our geocasting protocol when we have only global updates started by sinks periodically and local updates triggered by mobile sensors. Global updates are sent from sinks with a frequency varying from 3 to 25 seconds. In simulations with only global updates, the queries are sent $100ms$ before the next global update starts. In simulations with construction of geocast structure followed by only local updates, the queries are sent every 10 seconds. As described in Section 4.7, mobile nodes send beacon packets periodically having a frequency based on the speed of the node and the mobile node beacon constant, k, which varies from 0.05 to 0.20.

4.8.3 Effect of Frequency of Global Updates

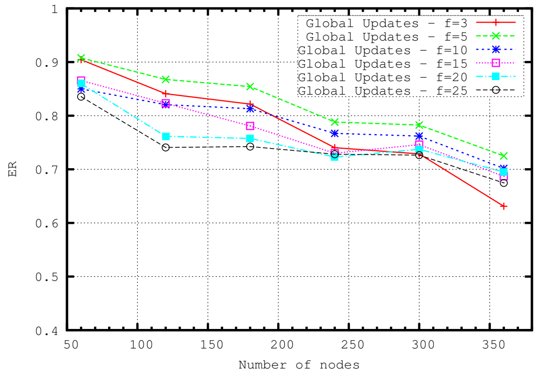
To estimate the most efficient global update frequency, we test different frequency values (i.e. f) varying from 3 to $25s$. In this set of simulations, sinks perform periodically global updates to update the routing tree and local convex hulls. There is no local update during the simulations, i.e. mobile nodes do not sent local update messages to their neighbors. Higher frequency of global updates means that the routing trees of sinks will be updated more frequently resulting in fresh geocast structure most of the time. The queries are sent just before the next global update starts to see the worst case behavior of different update frequency values.

In Figure 4.21(a), we evaluate ERs of different f values when we increase the number of nodes and the number of mobile nodes in the network. We observe that increase in number of nodes and number of mobile nodes in the network results in decreases of ERs for different global update frequencies. The best ER is obtained when $f = 5$. Although global updating every 3 seconds refreshes the tree structure more frequently, the drop in the ER of $f = 3$ is very large when we increase the number of nodes. This is mainly due to the overlapping of global updates. Overlapping means that before finishing one global updating, the next one starts. The overlapping time gets longer when we have more nodes in the network. Therefore, for bigger network sizes 3 seconds is not enough to finish one global updating. For instance, for a network having 360 nodes, it takes around 3.75 seconds to finish one global update. For other f values bigger than 5, ER is worse than ER of $f = 5$ for the given network sizes. It is important to point out that for networks bigger than 360 nodes, 5 seconds may also not enough to finish one global updating. However, it is the best frequency for the range of network sizes we consider here.

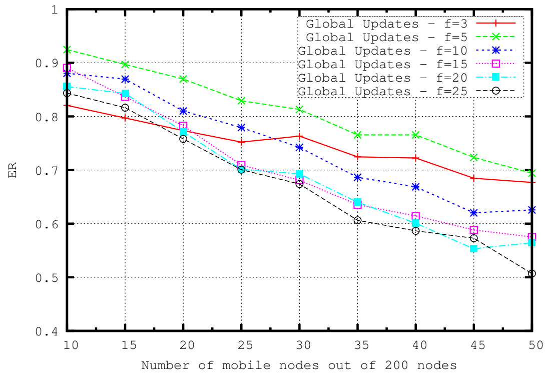
In Figure 4.21(b), which reflects the results of *Scenario 2*, we plot the dependency between ER and the number of mobile nodes. While in *Scenario 1* (Figure 4.21(a)), the ratio of mobile nodes to the number of nodes in the network is constant, here we vary the number of mobile nodes in the network and keep the total number of nodes fixed. ER of $f = 5$ is also the best in this scenario. In general, when we have higher mobility (i.e. more mobile nodes) in the network, the ER decreases for all values of f.

Finally, in Figure 4.21(c), we explore the influence of speed of mobile nodes on ERs of different update frequencies. The frequency $f = 5$ again achieves the best performance. However, different than the other scenarios, ERs first slightly decrease when we increase the speed of mobile node up to $9m/s$, then ERs stay more or less stable or slightly increase for the speeds higher than $9m/s$ in this scenario. This behavior is due to the adaptation of timeout mechanism (i.e. *timeout interval* is calculated by Equation (4.2)) according to speed of mobile nodes. For higher speeds, timeout interval is shorter; hence, up-to-date convex hull descriptions can be obtained quicker. Since the geocast structure is updated more frequently for higher speeds by the timeout mechanism, ER results are the same or better for higher speeds.

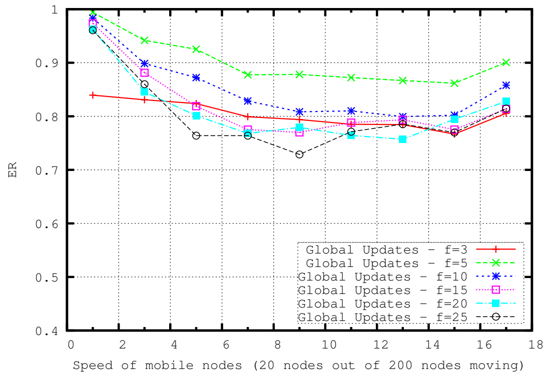
Maintaining up-to-date convex hull information by exchanging control packets evokes



(a) *Scenario 1:* Impact of Number of Nodes

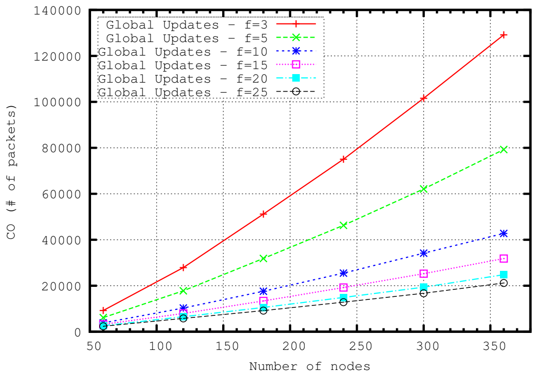


(b) *Scenario 2:* Impact of Number of Mobile Nodes

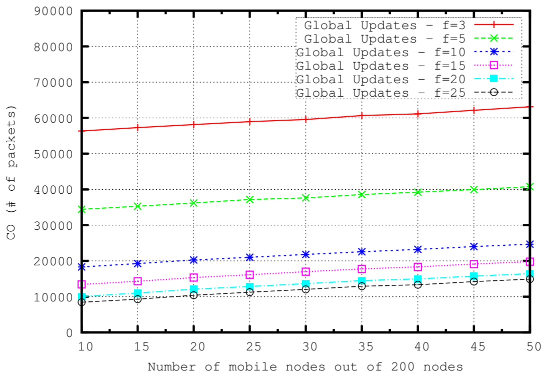


(c) *Scenario 3:* Impact of Mobile Node Speed

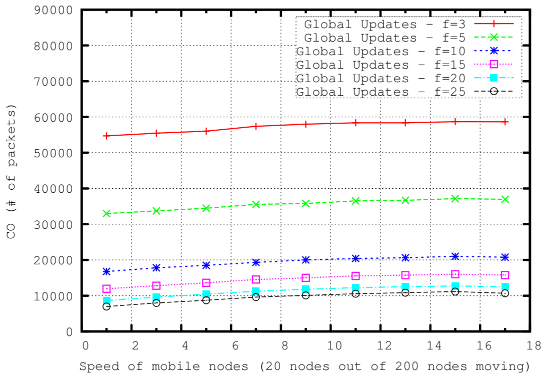
Figure 4.21: *Execution Ratio for different frequencies of global updates*



(a) *Scenario 1:* Impact of Number of Nodes



(b) *Scenario 2:* Impact of Number of Mobile Nodes



(c) *Scenario 3:* Impact of Mobile Node Speed

Figure 4.22: *Communication Overhead for different frequencies of global updates*

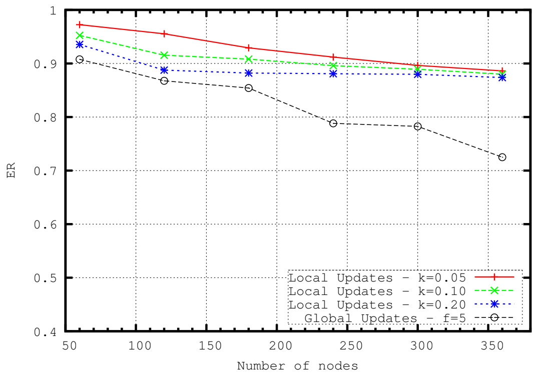
control overhead. Possessing the most up-to-date tree structure provides better chance to build a more efficient query dissemination solution. However, there is a trade-off with the update interval: if trees are updated too often, the benefit of having current state information diminishes and most of the energy will be consumed exchanging control packets (i.e. hello, convex hull update messages). In what follows, we investigate the communication overhead of different update frequencies in the three scenarios.

Figure 4.22(a) presents the results in terms of communication overhead when we vary the number of nodes. CO values of global updating with $f = 10, 15$, and 20 are very close to each other. On the other hand, when we increase frequency to 5 and 3, the increase in the communication overhead is very large with increasing number of nodes. The trend in Figure 4.22(a) is steeper when we increase the number of nodes; on the other hand, in Figure 4.22(b) and Figure 4.22(c), the graphs increase gradually. In Figure 4.22(b), we observe that COs of all frequencies slightly increase when we increase the number of mobile nodes in the network. The reason for this behavior is that more convex hull update messages are generated due to higher mobility rate. Figure 4.22(c) shows the CO results with increasing mobility speed. The figure has a very similar shape with the varying number of mobile nodes graph in Figure 4.22(b). Again, due to adaptation of timeout interval (i.e. Equation (4.2)) according to the speed of `MN`, higher speed results in more convex hull update messages in the network. However, increasing network size is problematic in terms of CO of global updates, whereas mobility rate and speed are not.

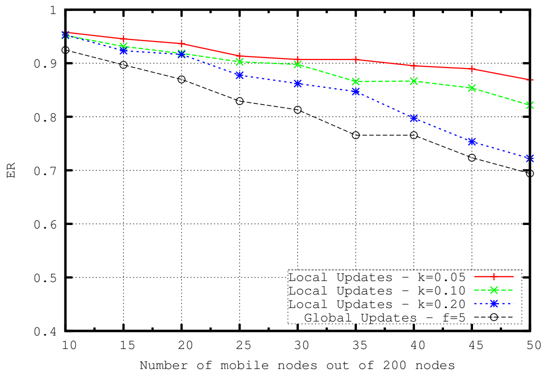
4.8.4 Comparison of Global and Local Updates Performance

In this set of simulations, only one global update (i.e. sinks helloing) is performed at the beginning of the simulation to build a full geocast structure (i.e. the routing trees and convex hulls) in the network. Then, only local updates from mobile nodes are sent in their neighborhoods to keep geocast structure up-to-date during the simulation. Performance of local updating with different k values are compared. Local updating is also compared with the best frequency of global updating. The parameter k affects the frequency of local updates performed by MNs. A small k results in triggering local updates more frequently. Assuming a mobility speed of $10m/s$ and $250m$ transmission range, when we have $k = 0.10$, `MN` sends a beacon packet for every $25m$ displacement.

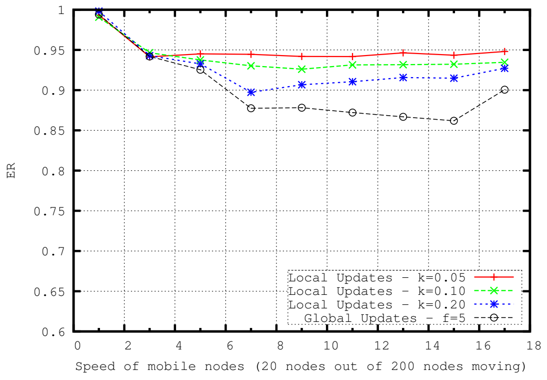
In Figure 4.23(a) we compare the local updates with different `MN` beacon constant, k, values and global updating with a frequency of 5 seconds which achieves the best performance as shown in the previous set of simulations. It is observed that ER of all local updates with different k values is better than the best global update. When we increase the number of nodes in the network by keeping the deployment area fixed, the density of the network also increases. This increase results in shorter distances between parent and child nodes. Hence, as observed in the figure, for bigger number of nodes, the ER values are very close to each other for different values of k because longer `MN` beacon intervals still achieve on-time updating of geocast structure in dense networks. With this observation, we can redefine k as *density factor* since it should be determined by the density of the network. In Figure 4.24(a), we compare the communication overhead of local and global updating. When sensor nodes are more densely deployed in the sensor field, the CO increases for both global and local updating. However, the CO of local updating with different k values is always lower than that of global updating. This is an expected results since local updates only reshape the branches



(a) *Scenario 1:* Impact of Number of Nodes

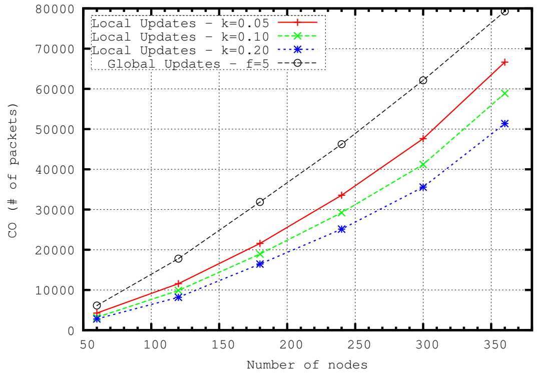


(b) *Scenario 2:* Impact of Number of Mobile Nodes

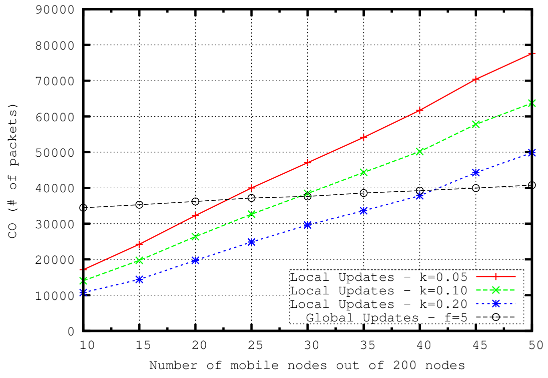


(c) *Scenario 3:* Impact of Mobile Node Speed

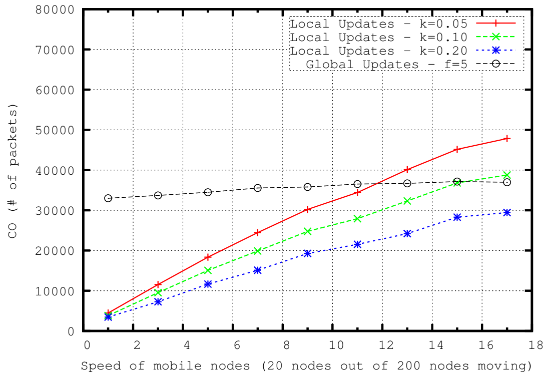
Figure 4.23: *Comparison of local and global updates in terms of Execution Ratio*



(a) *Scenario 1:* Impact of Number of Nodes



(b) *Scenario 2:* Impact of Number of Mobile Nodes



(c) *Scenario 3:* Impact of Mobile Node Speed

Figure 4.24: *Comparison of local and global updates in terms of Communication Overhead*

affected by mobile nodes so the messaging is reduced in the local updating.

Figure 4.23(b) plots the ERs of local and global updating when we increase the number of mobile nodes in the network. Local updating with $k = 0.05$ gives the best results of ER. While ERs of local updating with $k = 0.05$ are very close to local updating with $k = 0.10$ and $k = 0.20$ for networks having less than 35 mobile nodes, this behavior changes for networks having more mobile nodes. Results show that when we have more mobile nodes, more frequent updates are needed to reflect the actual geocast structure. For any k value of local updating, global updating gives worse ER results than local updating. Figure 4.24(b) presents the results in terms of communication overhead when we vary the number of mobile nodes. As expected if we have more number of mobile nodes, messaging, so communication overhead increases for local updating. On the other hand, CO slightly increases for global updating when we increase the number of mobile nodes since communication overhead of global updating mainly depends on the number of nodes in the network and some convex hull update messages due to timeout mechanism. However, as we observed from Figure 4.23(b) and Figure 4.24(b), it is possible to determine a k value for local updating which results in less CO and still performs better than global updating in terms of ER. If we compare the COs of local and global updating, for example for $k = 0.20$, when we have 40 mobile nodes in the network, COs of local and global updates are more or less the same. This number is getting smaller for smaller k values (i.e. COs of local and global updates are similar when we have 25 nodes for $k = 0.05$ and 30 nodes for $k = 0.10$) of local updating because smaller k values create more beacon and update messages in the network.

In Figure 4.23(c), we evaluate ERs of local and global updating when we increase mobility speed. ERs of local updating with different k values first slightly decrease when we increase the speed of mobile node upto $9m/s$, then ERs slightly increase or stay unchanged for the speeds higher than $9m/s$. For higher speeds of mobile nodes, `MN_BEACON_INTERVAL` gets shorter and mobile nodes become the leaf nodes of the trees more quickly since they send leave messages to their child nodes quickly after they start to move. In Figure 4.24(c), we explore the influence of speed of mobile nodes on COs of local and global updates. When we increase the speed of mobile nodes, COs of local updates with different k values increase. This is due to the fact that higher speed means shorter `MN_BEACON_INTERVAL`; thus, more `MN` beacon and convex hull update messages for local updating mechanism. On the other hand, CO of global updating slightly increase with increasing speed. CO of local updating with $k = 0.20$ is always smaller than CO of global updating; however, when we assign $k = 0.10$, the COs of local and global updating are very similar for the speed of $15m/s$ (i.e. around $55km/h$).

4.8.5 Discussion of Simulation Results

As we observed from the simulation results, local updates perform very well in terms of ER; on the other hand, local updating is much more energy efficient than global updating in networks having any number of nodes, but, small number of mobile nodes (i.e. around 10-15% of the number of nodes) moving with a speed varying from low to high (i.e. around 1-$20m/s$). This specification is very much valid for mostly static sensor network scenarios in which most of the sensors are static and some sensors are attached to people or vehicles such as firefighters or firetrucks, unmanned aerial vehicles (UAVs) moving at low or medium velocities in an Emergency Response Application.

It is possible to apply some improvements to local updating mechanism if we assume that the mobile nodes are aware of their direction of movement. Some of the acknowledgement packets can be eliminated when a mobile node knows whether it moves towards the sink of the tree or in the opposite direction. If mobile node moves towards sink, there is no need to have acknowledgements from the neighbor nodes in the same hop count level. Since the number of acknowledgement packets increases as the network size gets bigger (i.e. the number of neighbors increases), reducing unnecessary acknowledgement packets will have a positive impact on the overall communication overhead. Mobile nodes can easily determine their movement direction, if they are aware of the position of the sink node.

4.9 Conclusion

We presented a protocol for routing of queries in a region of interest in this chapter. The protocol relies on a structure that consists of routing trees and convex hulls of tree branches and the full trees, which represent the respective coverage areas. This structure is used for an efficient routing of queries. We compared our geocasting protocol, namely GeoCHT, with GEAR. The comparison was done for both static and mobile scenarios in terms of routing accuracy and on networking. In static scenarios, the execution ratios of both protocols are very close to each other and change between 90% and 100%. GEAR has a slightly better execution ratio than GeoCHT, while GeoCHT outperforms GEAR in the other metrics. False execution ratio is zero for both protocols. False injection ratio of GeoCHT is more or less constant and lower than GEAR for most of network configurations. The query delay and network load of GeoCHT are much lower than in GEAR. There are similar results from the mobile scenarios. The execution ratio of GEAR is slightly better than of GeoCHT, both staying in the range of 90% and 100%. GeoCHT outperforms GEAR considerably in the other metrics, false injection ratio, query delay and network load, for all the considered mobility scenarios. False execution ratio is again zero for both protocols. Since both GEAR and GeoCHT assume that nodes have an estimate of their positions, we have also tested and compared the two protocols to observe the effect of positioning error. The execution ratios of both protocols decrease with a very similar slope when we increase the localization error. The query delay and network load of GeoCHT are unaffected by the positioning error. They are both lower than in GEAR, where we also see that the delay increases with the increase of the positioning error.

In the second part of the chapter, we focus on handling mobility where several sensors move in the sensor field. The mobility imposes high needs in the updating of the geocast structure. We proposed a local update mechanism, where updates are triggered by moving sensor nodes, and are kept constrained only to the parts of the structure affected by those nodes. We compared the performance of local updates with global updates of the structure in terms of query execution ratio and the communication overhead caused by the two types of updates. The proposed local updating mechanism is designed for coping well with such dynamic environments, by using distributed local link reversal. The simulation results demonstrate that geocasting with local updating performs a better query dissemination in terms of execution ratio with relatively low communication overhead, compared to geocasting with global updating. Although increasing number and speed of mobile nodes results in increase in the overall communication overhead on the network, local updating still performs better than global updating in terms of execution ratio. This is due to the fact that local updating performs continuous updating of the tree branches which are affected by the mobile nodes. The most critical issue is the mobility rate (i.e. number of mobile nodes / total number of nodes) to decide which approach performs better. Indeed, if the network under consideration is highly mobile where most of the nodes are moving, tree based routing approaches becomes very costly to maintain the routing structure. Although tree based routing protocols can tolerate topology changes in mostly static sensor network deployments, they cannot survive excessive topology changes in highly mobile deployments.

CHAPTER V [*]

Data Dissemination in Mobile Multi-sink Wireless Sensor Networks

A new category of intelligent sensor network applications emerges where motion is a fundamental characteristic of the system under consideration. In such applications sensors are attached to vehicles, animals or people that move around large geographic areas. For instance, in mission critical applications of wireless sensor networks, sinks can be associated to first responders such as firefighters, but also to vehicles like unmanned aerial vehicles (UAVs), or in a road safety application, sensors are carried by cars in cities. In such scenarios, reliable data dissemination of events is very important, as well as the efficiency in handling the mobility of both sinks and event sources. In this chapter we propose a virtual infrastructure and a data dissemination protocol exploiting this infrastructure, which considers dynamic conditions of multiple sinks and sources. The architecture consists of 'highways' in a honeycomb tessellation, which are the three main diagonals of the honeycomb where the data flow is directed and event data is cached. The highways act as rendezvous regions of the events and queries. Once a query is issued, it is sent to one of the highways and searches relevant data stored in the highway. When the data is found, it is sent to the sink which has issued the query. Our protocol, Hexagonal cell-based Data Dissemination (HexDD) is fault-tolerant, meaning it can bypass holes in the network. We analytically evaluate the communication cost and hot region traffic cost of HexDD and compare it with other approaches.

[*]This chapter is a minor revision of the paper with the title "A virtual infrastructure based on honeycomb tessellation for data dissemination in multi-sink mobile wireless sensor networks" In EURASIP Journal on Wireless Communications and Networking, Submitted in April 2011 [6] and some parts of the chapter also appear in papers published with the titles "Data dissemination of emergency messages in mobile multi-sink wireless sensor networks" In the Proceedings of the 9th IFIP Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net), June 2010 [1], and "A fault-tolerant data dissemination based on Honeycomb Architecture for Mobile Multi-Sink wireless sensor networks" In the Proceedings of 6th International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), December 2010 [5].

5.1 Introduction

In all applications of wireless sensor networks, the primary goal is to collect useful information by monitoring phenomena in the surrounding environment. Common sensing tasks are heat, pressure, light, sound, vibration, presence of objects, etc. In WSNs, each sensor individually senses the local environment, but collaboratively achieves complex information gathering and dissemination tasks. Typically a WSN follows the communication pattern of convergecast, where sensors -*source* nodes- generate data about a phenomenon and relay streams of data to a more resource rich device called *sink*. This procedure is called *data dissemination*, which is a preplanned way of distributing data and queries of sinks among the sensors. In this chapter, we focus on data dissemination in *mobile multi-sink* deployments of WSNs.

The work presented in this chapter is mainly motivated by disaster management scenarios (see Chapter 2.4) where we have a mobile multi-sink wireless sensor network in which the deployment of sensors is performed in a random fashion, e.g., dropping sensors from helicopters flying above the field [3]. As shown in Figure 2.3, in this mobile multi-sink WSN, UAVs, emergency responders, e.g. firefighters, or vehicles, e.g. firetrucks, carry *sink* nodes on-board. These mobile sinks are used to collect more reliable data about the event in the dangerous/inaccessible regions. In this scenario, both the number of sources and that of mobile sinks may vary over time. The speed of sources and sinks also vary from a typical pedestrian to a flying UAV.

Sink mobility brings new challenges to data dissemination in wireless sensor networks. Figure 5.1 shows the movement trace[†] of a UAV carrying a sink on-board. Since the location of the sink changes in time, as shown in the figure, the difficulty for sensor nodes is to efficiently track the location of the mobile sink to report the collected measurements about the event. Although several data dissemination protocols have been proposed for sensor networks, e.g. Directed Diffusion [85], they all suggest that each mobile sink needs to periodically flood its location information through the sensor field, so that each sensor is aware of the sink location for sending future events and measurements. However, such a strategy leads to increased congestion and collisions in the wireless transmission and is thus mainly suited for (semi) static setups.

As flat networks and flooding-based protocols do not scale due to frequent location updates from multiple sinks, overlaying a virtual infrastructure over the physical network has been investigated as an efficient strategy for data dissemination towards mobile sinks [76]. In this chapter, we investigate the use of virtual infrastructures to support mobile sinks in WSNs. Once a virtual infrastructure is overlaid over the physical network, it acts as a *rendezvous* region for storing and retrieving collected measurements. Sensor nodes in the rendezvous region store the generated data during the absence of the sink. When the mobile sink crosses the network, the sensors in the rendezvous region are queried to notify of the event data.

We first present the advantages and challenges of using mobile sinks in WSNs. Next, we introduce the *Honeycomb Architecture* and the protocol based on it, *Hexagonal cell-based Data Dissemination (HexDD)*. HexDD is a geographical routing protocol based on this virtual infrastructure concept, proposing rendezvous regions for events (data caching) and queries (look-up). It is designed to improve network performance in terms of data delivery

[†]The movement trace data collected in the real-world AWARE experiments in May 2009 in Utrera, Spain.

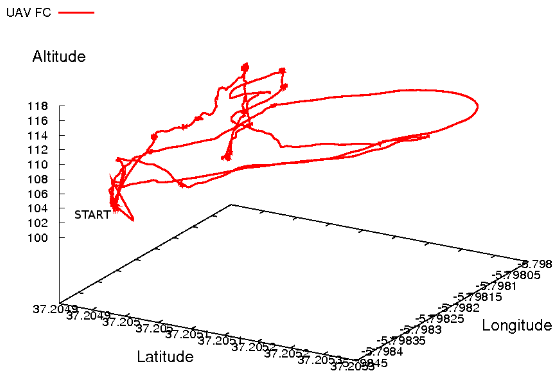


Figure 5.1: *UAV mobility trace from AWARE experiments*

ratio and latency, besides meeting the traditional requirements of WSNs, such as energy efficiency.

Our contribution is a data dissemination protocol with a fault-tolerance mechanism that does not require additional networking overhead, such as extra messaging to find alternative paths. In contrast to the rich literature on virtual infrastructure based data dissemination, especially those using greedy forwarding to send data from sources to rendezvous region, data sent by sources is forwarded along predefined regions called *highways*, which are the rendezvous regions in HexDD. The followings are some of the other key highlights of this work:

(i) We discuss the advantages and challenges of mobile sinks and present a review of existing virtual infrastructure based data dissemination protocols for mobile multi-sink WSNs.

(ii) We propose HexDD protocol to accommodate the dynamics of the WSN due to stimulus and sink mobility, in such a way that it avoids excessive updates caused by a frequently changing environment. HexDD also provides a fault-tolerance mechanism that detects routing holes and establishes alternative forwarding paths.

(iii) We analytically evaluate the communication cost and hot region traffic cost of HexDD and compare it with other approaches.

(iv) We evaluate the performance of HexDD with extensive simulations in NS2, and present a large study of comparisons with two other virtual infrastructure based protocols (see Chapter 6.1.2). The protocols with different virtual infrastructures allow us to study the effects of the virtual infrastructure shape and the data dissemination strategy on the networking performance.

(v) We show the hot spot regions (i.e., heavily loaded nodes around rendezvous areas) that are created by different virtual infrastructure based protocols. We present a method for resizing of rendezvous region in HexDD to alleviate hot spot problem in the network (see Chapter 6.1.2).

The rest of this chapter is organized as follows: The related works are introduced with their strengths and weaknesses in Section 5.2. Section 5.3 introduces the honeycomb architecture and HexDD protocol. Section 5.4 provides analytical studies of communication cost and hot spot traffic cost of HexDD. Finally, Section 5.5 draws the conclusions. Besides the analytical evaluation of HexDD protocol in this chapter, in Chapter 6 we perform an extensive comparative simulation study of HexDD protocol in two different WSN applications, i.e. mostly static and highly mobile.

5.2 Related Work

5.2.1 Mobility patterns and data collection strategies

Sink mobility can be classified as *uncontrollable* or *controllable* in general as we discussed in Chapter 2.2. The former is obtained by attaching a sink node on a mobile entity such as an animal or a shuttle bus, which already exists in the deployment environment and is out of control of the network. The latter is achieved by intentionally adding a mobile entity e.g., a mobile robot, into the network to carry the sink node. In this case, the mobile entity is an integral part of the network itself and thus can be fully controlled [102].

Different sink mobility patterns provide different data gathering mechanisms ranging from single hop passive communication (i.e., direct-contact data collection), which require controllable sink mobility, to multi-hop source to sink solutions, which can be achieved by uncontrollable or controllable sink mobility.

Direct-contact data collection has great advantage for energy savings. That is, sinks visit (possibly at slow speed) all data sources one by one and obtain data directly from them. This data collection strategy needs intelligent sink movement computed as the best sink trajectory that covers all data sources and minimizes data collection delay [142]. With this approach, maximum energy efficiency and longest network lifetime is achieved at the expense of very long delays. This mobility scheme is feasible for delay tolerant applications.

Rendezvous-based data collection is proposed to achieve a good trade off between energy consumption and time delay. Sensors send their measurement to a subset of sensors called *rendezvous points* (RPs) by multi-hop communication; a sink moves around the network and retrieves data from encountered RPs. The use of RPs enables the sink to collect a large volume of data at energy cost of multi-hop data communication at a time without traveling a long distance and thus greatly decreases data collection delay. If the virtual infrastructure of rendezvous-based protocol is well designed, one can achieve scalability and energy efficiency. Rendezvous-based data collection can be used when we have uncontrollable (e.g., random or fixed-track) sink movement in a WSN.

5.2.2 Data dissemination protocols

Several data dissemination protocols have been proposed for WSNs with mobile sinks. The proposed protocols fall in two major categories: (i) *Flooding based* and (ii) *Virtual infrastructure based*. In general, virtual infrastructure-based protocols can be divided into (i) *backbone-based* approaches (e.g. [47]), and (ii) *rendezvous-based* approaches (e.g. [150])

depending on how the virtual infrastructure is formed by the set of potential storing nodes. All protocols discussed in this section assume uncontrolled mobility in the network.

Directed Diffusion [85] is a flooding-based approach introducing data-centric routing for sensor networks. In this approach, each sink must periodically flood its location information through the sensor field. This procedure sets up a gradient from a sensor node to the sink node so that each sensor becomes aware of the sink's location for sending future data. Although directed diffusion solves the problem of energy-efficiency by using several heuristics to achieve optimized paths, its flooding-based approach does not scale with the network size and increases the network congestion.

PEG (Pursuit-Evasion Games) [149] is a sensor network system that detects an uncooperative mobile agent, *evader*, and assists an autonomous mobile robot called the *pursuer* in capturing the evader. The routing mechanism used in PEG, namely *landmark routing*, uses the node at the center of the network as landmark (i.e. only one rendezvous point) to route packets from many sources to a few sinks. It constructs a spanning tree having the landmark node as the root of the tree. For a node in the spanning tree to route an event to a pursuer, it first sends the data up to the root, the landmark. The landmark, then, forwards the data to the pursuer. The pursuer periodically informs the network of its position by picking a node in its proximity to route a query to the landmark. Since data dissemination used in PEG is a combination of directed diffusion [85] towards the landmark and central re-dissemination, in order to build the gradients from sensors to landmark node (i.e. spanning tree), it uses flooding-based approach (i.e. each node sends a beacon packet which is further re-broadcasted by all the neighbors of the node) which results in broadcast storm problem increasing the congestion.

As the flat networks and flooding-based protocols do not scale, overlaying a virtual infrastructure over the physical network often has been investigated as an efficient strategy for data dissemination in mobile WSNs [76]. This strategy uses the concept of virtual infrastructure, which acts as a rendezvous area for storing and retrieving the collected measurements. The sensor nodes belonging to the rendezvous area are designated to store the generated measurements during the absence of the sink. After the mobile sink crosses the network, the designated nodes are queried to report the sensory input. The concept of overlaying a virtual infrastructure over the physical network has several advantages. The infrastructure acts as a rendezvous region for the queries and the generated data. Therefore, it enables the gathering of all of the generated data in the network and permits the performing of certain data optimizations (e.g. data aggregation) before sending the data to the destination sink [76]. Secondly, in WSNs deployed in harsh environments, source nodes can be affected by several environmental conditions (e.g., wildfire, etc.), and therefore, the risk of losing important data is high. To ensure the persistence of the generated data, the source node can disseminate the data towards the rendezvous area instead of storing it locally. Thus, the virtual infrastructure enables data persistence against node failures. Main disadvantage of using a virtual infrastructure is the creation of hotspot regions in the network. However, it is possible to solve this problem by adjusting the size of rendezvous regions. Several protocols that implement a rendezvous-based virtual infrastructure have been proposed in the literature. They vary in the way they construct the virtual infrastructure. In the rest of this subsection we summarize these protocols.

The Geographic Hash Table (GHT) [139], which is illustrated in Figure 5.2(a), introduces

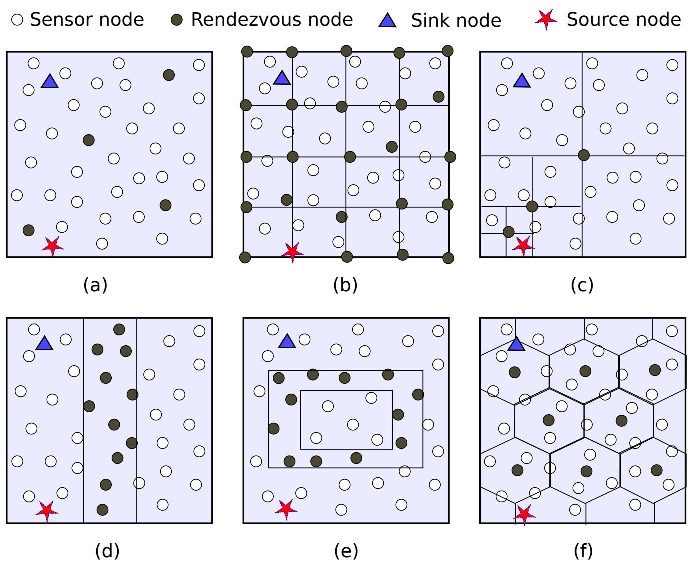


Figure 5.2: *Virtual infrastructure-based data dissemination protocols: (a) GHT (hashed location), (b) TTDD (grid structure), (c) QDD (quad-tree structure), (d) LBDD (line-based structure), (e) RailRoad (rail structure), (f) GCA (hexagonal clustering)*

the concept of data-centric routing and storage. GHT hashes keys into geographic coordinates, and stores a key-value pair at the sensor node geographically nearest the hash of its key. In GHT, the data report type is hashed into geographic coordinates, and the corresponding data reports are stored in the sensor node, called *home-node*, which is the closest to these coordinates. This *home-node* acts as a rendezvous node for storing the generated data reports of a given type. There are as many *home nodes* as data types. The main drawback of this approach is the hotspot problem because all data reports and queries for the same meta-data are concentrated on the same *home node*. This may restrict the scalability and the network lifetime.

In Two-tier Data Dissemination (TTDD) [173], each source node proactively builds a uniform virtual grid structure throughout the sensor field, as shown in Figure 5.2(b). A sink floods a query within its local grid cell. The query packet then propagates along the grid to reach the source node. While the query is disseminated over the grid, a reverse path is established towards sink and data is sent to the sink via this reverse path. If the stimulus is mobile, number of sources and grids increase. This situation can lead to excessive energy drain, and therefore, limit the network lifetime.

Quadtree-based Data Dissemination (QDD) [120] protocol defines a common hierarchy of data forwarding nodes created by a Quadtree-based partitioning of the physical network into successive quadrants, as shown on Figure 5.2(c). In this approach, when a source node detects a new event, it calculates a set of rendezvous points by successively partitioning the sensor field into four quadrants, and the data reports are sent to the nodes which are closer to the centroid of each successive partition. The mobile sink follows the same strategy for the query packet transmission. The main drawback of this approach is that few static nodes will be selected as rendezvous points inducing a hot spot problem which may decrease the network lifetime and reliability.

Line-based Data Dissemination (LBDD) [75], which is proposed for mobility of sink and source nodes, defines a vertical *line* or *strip* that divides the sensor field into two equal sized parts, as shown in Figure 5.2(d). Nodes within the boundaries of this wide *line* are called *inline nodes*. This virtual line acts as a rendezvous area for data storage and look-up. When a sensor detects a new event, it transmits a data report towards the nodes in the virtual line. This data is stored on the first inline node encountered. To collect the generated data reports, the sink sends its query toward the rendezvous area. This query is flooded along the virtual line until it arrives to the inline node that owns the requested data. From there the data report is sent directly to the sink using greedy forwarding. However, using a line as rendezvous area at the middle of the network can results in high latency for the nodes near the boundary of the network.

RailRoad [150] places a virtual *rail* in the middle of the deployment area, as shown in Figure 5.2(e). When the source node generates data, the generated data is still stored locally, but corresponding meta-data (i.e. event notification) is also forwarded to the nearest node inside the rail. When a sink node wants to collect the generated data, a query message is sent into the rail region. This message travels around the rail. When it reaches the rail node that stores the relevant event notification, the rail node sends a query notification message to the source node. Finally, source node sends data directly to the sink using greedy forwarding.

Geographical Cellular-like Architecture (GCA) [47], which is a backbone-based approach, defines a hierarchical hexagonal cluster architecture that basically adopts the concept of home-agent used in cellular networks. Each cluster is composed of a *header* positioned at the center of the hexagonal cell and *member sensors*, as presented in Figure 5.2(f). The mobile sink sends its query to the cell header that sink belongs to. The query packet then is propagated to all cell headers. When the sink moves to another cell, it registers to the new cell's header and also informs its old cell header (home-agent) about its new header's position. The data packets still are propagated towards the home-agent, which further forwards the packet to the sink's new header. In case of sink mobility, GCA results in inefficient (non-optimal) routing path which may increase the data delivery latency.

The hierarchical cluster-based data dissemination protocol (HCDD) [106] defines a hierarchical cluster architecture to maintain the location of mobile sinks and find paths for the data dissemination from the sensors to the sink. Unlike GCA, HCDD does not require powerful position aware nodes. Each cluster is composed of a cluster head, several gateways, and ordinary sensors. When a mobile sink crosses the network, it registers itself to the nearest cluster head. Then a notification message is propagated to all cluster heads. During this procedure, each cluster head records the sink ID and its sender such that the transmission of future data reports can be performed easily from sources to sink.

Table 5.1 shows a classification of the existing data dissemination protocols, which support multiple, mobile sinks and how HexDD differs from these existing works. All *rendezvous-based* approaches use greedy geographic routing (i.e. greedy forwarding, GF). Greedy geographic routing is attractive in wireless sensor networks due to its efficiency and scalability. However, greedy geographic routing may incur long routing paths, and even fail due to routing holes on random network topologies. Most of the previous works do not discuss how

Table 5.1: *Comparisons of virtual infrastructure based data dissemination protocols for WSNs with mobile sinks*

Year	**GHT** (2002)	**TTDD** (2002)	**QDD** (2006)	**LBDD** (2008)	**RailRoad** (2005)	**GCA** (2005)	**HCDD** (2006)	**HexDD** (2010)
Position awareness	+	+	+	+	+	+	-	+
Virtual infrastructure	hashed location	grid	quad-tree	line/strip	rail	clusters	clusters	highways
Disseminated information	data	data	data	data	meta-data	data	data	data
Data reports location	1 node	1 node	1 out of N nodes	1 out of N nodes	1 out of N nodes	1 node	1 out of N nodes	1 out of N nodes
Routing towards RPs	GF	GF	GF	GF	GF	GF	tree-based routing	cell-address based routing
Routing hole recovery	-	-	-	-	+	-	-	+
Metric of interest	energy	energy	energy reliability	energy	energy	energy	energy	reliability latency energy

to maintain the virtual infrastructure if there are holes, a large space without active sensors, which is a common behavior in any real WSN deployment. To recover from the local minima, where a node does not have a neighbor closer than itself to the destination, GPSR [89] and GOAFR [97] route a packet around the faces of a planar subgraph extracted from the original network, while limited flooding is used in [155] to circumvent the routing hole. Unfortunately, the recovery mode inevitably introduces additional overhead and complexity to geographic routing algorithms. The main problem of the *backbone-based* approach is the need to maintain the structure. In addition, the hotspot problem may occur as the traffic is concentrated over a group of cluster headers.

Since the previous works do not provide efficient fault-tolerant data dissemination for emergency response scenarios, we propose a novel rendezvous-based data dissemination protocol which uses hexagonal cells for geographic routing to support sink mobility. To bypass routing holes, we present a simple hole recovery mechanism which avoids to flood any other control message to find new bridge nodes. The hole recovery mechanism tries to find the shortest path to recover holes; therefore, it decreases latency and increases reliability of the data dissemination. Also, in WSNs where there is no hole, the proposed protocol achieves a high data delivery ratio and low data delivery delay and energy consumption.

5.3 Honeycomb Architecture and Dissemination Protocol

In this section we describe how the physical network is partitioned into virtual hexagonal cells by the Honeycomb Architecture (see Figure 5.5), and how this architecture is employed by the geographical routing HexDD. Individual sensor nodes in the network are bound to cells of the virtual hexagonal tessellation based on their geographic locations. The architecture also defines three principle diagonal lines – highways (or border lines) – which divide the sensor field into six parts. The lines, which intersect at the center of the network, constitute the rendezvous region for queries and data.

Division of the sensor field into a regular tessellation is energy efficient compared to other schemes such as Voronoi diagram division [126]. The construction of a Voronoi diagram consumes high energy in resource constrained sensor nodes. Instead of square tessellation, which is used in many protocols [173], [170], we use a honeycomb tessellation for the homogeneous neighborhood it provides, i.e. all neighbors of a cell share an edge with the cell, no neighboring cells that share only a corner.

Hexagonal cells are used in literature for various applications [47, 148, 107]. Cellular phone station placement is one of the very well known applications of hexagonal tessellation. It is important to point out that in this chapter we do not use the concept of cellular networks. In cellular networks, a land area is divided into hexagonal cells having a base station located in middle to provide non-overlapping service to the entire network. In our proposal, we do not assume a clusterhead (i.e. data collector) at the middle of a cell. Here, we use hexagonal cells only for the purpose of geographical routing towards a region. Differently from [148], where the hexagonal grid defines the topology of the network, meaning a sensor node in each corner of the grid, we do not assume a regular topology but a random deployment.

Creating of the architecture and our routing protocol require knowledge of location. We assume that sensor nodes are location-aware and also know the network boundaries, as it is also assumed in [47]-[75]. The location information can be obtained either by GPS-free localization mechanisms [48, 59] or by means of a virtual coordinate system [138] during the network initialization phase. Two sensors can communicate when they are within a distance R of each other, called the communicable distance. We assume that the minimum radio range R_{min} is the same for all nodes. Through periodic interactions (beacon packets), a sensor node can learn the location and cell of its neighbors. Sensor nodes are mainly static, and there are multiple sinks moving randomly in the sensor field. Sinks are equal from the information point of view; it does not matter to which sink a data packet is sent.

In the following we introduce the operations of HexDD protocol. The first phase is Hexagonal Cell-Based Network Partitioning, which establishes the architecture, i.e. honeycomb cells and rendezvous areas are formed. This phase is performed in the network setup. After this setup, the network becomes ready to execute the Hexagonal cell-based Data Dissemination (HexDD) protocol.

5.3.1 Hexagonal Cell-Based Network Partitioning

Honeycomb Architecture overlays a virtual honeycomb over the sensor field as shown in Figure 5.3(a). In the honeycomb tessellation, each cell has six neighbors covering the surroundings from all directions. For two adjacent cells, every sensor node in one cell can communicate with all the nodes in the other cell. This defines the edge length of the hexagonal cell.

As illustrated in Figure 5.3(b), the longest distance between two adjacent cells is $l_{|AB|} = \sqrt{13}r$, where r is the edge length of the hexagon. In order for all nodes in two adjacent cells to be able to communicate with each other, the longest length must satisfy $l_{|AB|} = \sqrt{13}r \leq R_{min}$ where R_{min} is the transmission range. Therefore, we choose the edge length of the hexagon, $r = R_{min}/\sqrt{13}$, such that sensors in adjacent cells are within *communicable distance* of each other.

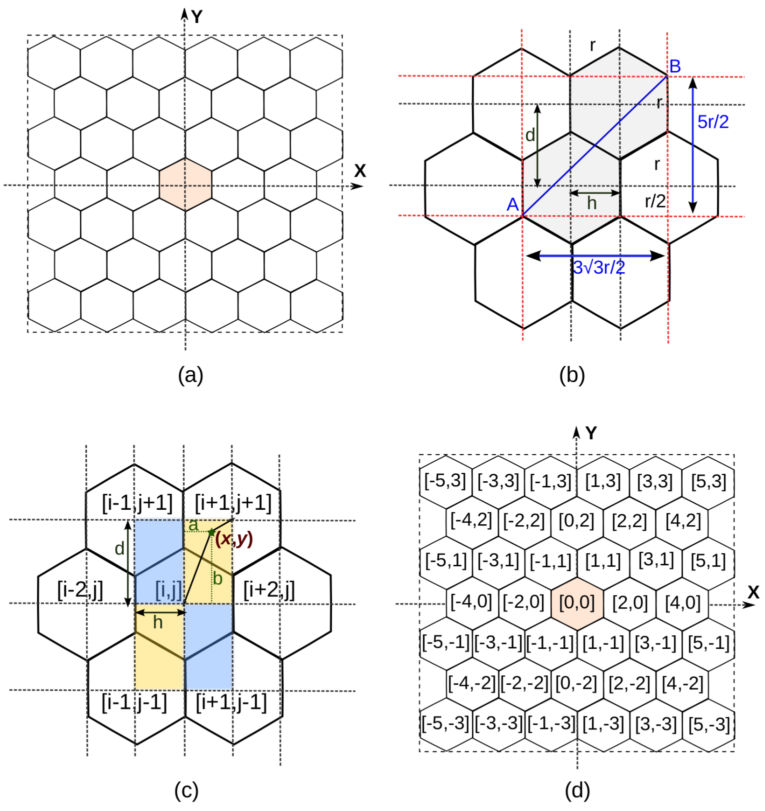


Figure 5.3: *(a) Hexagonal tessellation of the deployment area, (b) Cell structure, (c) Node-cell association, and (d) Cell Naming*

Node-Cell Association and Cell Naming

In the honeycomb architecture, a hexagonal cell placement and node association scheme need to be established. In this scheme, hexagonal virtual cells' central points are positioned according to Figure 5.3(c). Apparently, $d = \frac{3}{2}r$ and $h = \frac{\sqrt{3}}{2}r$, where r is the edge size of the hexagonal cell. Each virtual cell center is located at (id, jh) where i and j are integers. A virtual cell centered at (id, jh) is named as the cell $[i, j]$. Figure 5.3(c) show the cell $[i, j]$ and its neighboring cells with their associated names in the XY coordinate system. Figure 5.3(d) shows the cell naming in honeycomb architecture.

At the first step, with the given hexagonal edge length, r, each sensor node uses its location information to associate itself with a virtual cell having a name of $[i, j]$. For the node-cell association algorithm (see Algorithm 2), we have used a similar geometrical approach as in [107]. For a node positioned at point (x, y), let $i = \lfloor x/h \rfloor$ and $j = \lfloor y/d \rfloor$. If $i + j$ is even (i.e. the node is in the yellow rectangle in Figure 5.3(c)), the node is either in cell $[i, j]$ or in cell $[i + 1, j + 1]$; if $i + j$ is odd (i.e. the node is in the blue rectangle in Figure 5.3(c)), the node is either in cell $[i + 1, j]$ or in cell $[i, j + 1]$ depending on which center is closer. With

Algorithm 2 Node-Cell Association

```
Input: $r$: edge size of the hexagonal cell, $(x, y)$: coordinates of the node
Output: $[i, j]$ be the cell name assigned to the node at $(x, y)$
I. Calculate distance between (x,y) and candidate cells' centers
$d = 3 * r/2; h = sqrt(3) * r/2;$
$i = int(x/h); j = int(y/d);$
$a = x - (i * h); b = y - (j * d);$
II. Check which center is closer
if $(i + j)\%2 == 0$ then
    if $a^2 + b^2 \leq (d - b)^2 + (h - a)^2$ then
        $[i, j] \Leftarrow [i, j]$
    else
        $[i, j] \Leftarrow [i + 1, j + 1]$
    end if
else
    if $b^2 + (h - a)^2 \leq (d - b)^2 + a^2$ then
        $[i, j] \Leftarrow [i + 1, j]$
    else
        $[i, j] \Leftarrow [i, j + 1]$
    end if
end if
```

this algorithm each sensor node uses its coordinates to associate itself with a hexagonal cell. The algorithm is lightweight in computing: It has a total of 16 lines of code. A typical node would need to go through only 9 lines of code to find its cell. There is no communication overhead. Each node executes the algorithm locally.

Cell Addressing

Next, we transform the cell names of the form $[i, j]$ into special cell addresses of the form $[H, I]$. This addressing is used in the data dissemination.

Figure 5.4 shows the cell addressing in honeycomb architecture. We assign addresses of the form $[H, I]$ to each sensor in the same cell, where H is the shortest cell-count of the node from the origin cell and I denotes the index of the hop-H hexagonal cell. The index starts at the right side of *line b* in Figure 5.4 and increases in the counter-clockwise direction. Hence, the nodes in the first-hop cells are addressed as $[1, 0]$, $[1, 1]$,..., $[1, 5]$. Observe that nodes of the form $[H, .]$ are all located on the same hexagonal ring at distance H form the center cell. Since the number of cells on H^{th} hop hexagonal ring is $6 \cdot H$, the cell addresses range from $[H, 0]$ to $[H, 6H - 1]$.

With the diagonal lines l, b, and r in Figure 5.4, the area is divided into six slices called *hextants* (marked with roman numerals in the figure). To build $[H, I]$ addresses from $[i, j]$ naming, we use the transformation rules of Table 5.2. H is equal to absolute value of j in the second and fifth hextants, and $(|i| + |j|)/2$ in the other hextants. I is calculated based on the hextant and the value of H. This special addressing has some useful properties which allows simple calculations for the packet flow towards the rendezvous regions.

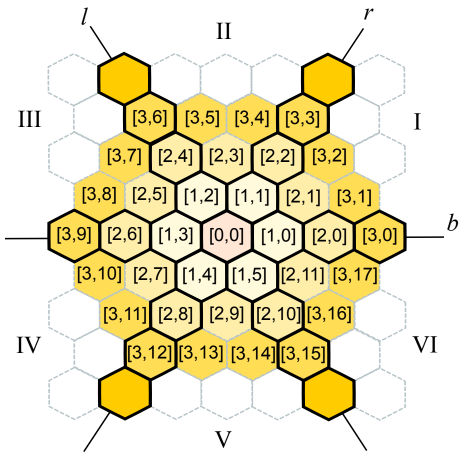


Figure 5.4: *Cell Addressing in Honeycomb Architecture*

Table 5.2: *Transformation rules from cell name [i, j] to cell address [H, I]*

Hextant	Condition	Transformation
1	$i > j, j \geq 0$	$[i, j] \Rightarrow [(\lvert i\rvert + \lvert j\rvert)/2, j]$
2	$\lvert i\rvert \leq \lvert j\rvert, j > 0$	$[i, j] \Rightarrow [\lvert j\rvert, 2H - (i + j)/2]$
3	$\lvert i\rvert \geq \lvert j\rvert, j > 0$	$[i, j] \Rightarrow [(\lvert i\rvert + \lvert j\rvert)/2, 3H - j]$
4	$\lvert i\rvert > \lvert j\rvert, j \leq 0$	$[i, j] \Rightarrow [(\lvert i\rvert + \lvert j\rvert)/2, 3H + \lvert j\rvert]$
5	$\lvert i\rvert \leq \lvert j\rvert, j < 0$	$[i, j] \Rightarrow [\lvert j\rvert, 5H + (i + j)/2]$
6	$i > j, j < 0$	$[i, j] \Rightarrow [(\lvert i\rvert + \lvert j\rvert)/2, 6H - \lvert j\rvert]$

In the honeycomb architecture, we classify the sensor nodes into two groups; (i) *border nodes* and (ii) *regular nodes*, according to their positions on the honeycomb tessellation.

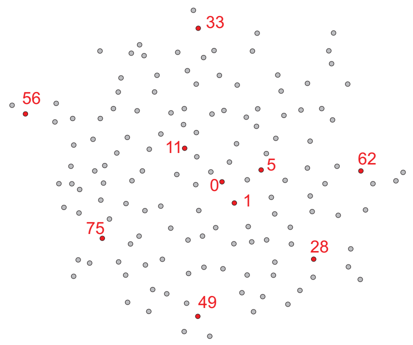
Definition 1: *All the cells addressed as* $[H, I]$ *are 'border cells' and fall on the diagonal lines if* $I = (k - 1) \cdot H$*, where integer* $k \in \{1, .., 6\}$*. The nodes associated with border cells are called 'border nodes'. All the other nodes are called 'regular nodes'.*

Figure 5.5 shows all the steps of the virtual cell formation and addressing in a 2000x2000 m^2 network having transmission range of 250 m. In the figure, some of the nodes are marked and their locations, cell names and cell addresses are given for illustration. The construction of this virtual infrastructure is carried out only once at the network setup stage.

Construction of Border Lines and Hextants

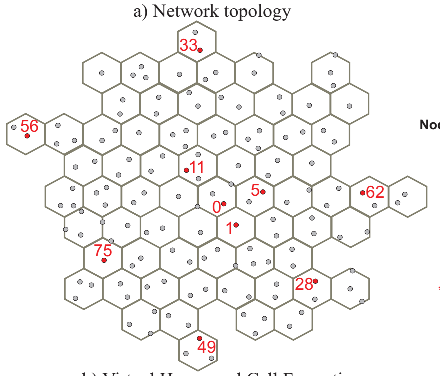
The honeycomb architecture defines three principle diagonal lines labeled as l, b, and r which are drawn through the origin of center cell, as illustrated in Figure 5.4. These lines divide the sensor field into six regions, called *hextants*. Each of six *hextants* is marked with roman numerals in the figure. More formally, all cells such that $(k - 1) \cdot H \leq I < k \cdot H$ belong to *hextant* k. A ring segment on H^{th} hop hexagonal ring in a hextant contains H cells.

All hexagonal cells on diagonal lines l, b, and r are borders of hextants, so called as *border cells*. These three diagonal lines act as rendezvous regions for data storage and look-up. Each half line, which starts from the origin, is the rendezvous area for the hextant which starts at this border line, assuming a counter-clockwise direction.



Node ID	Location
0	(1007, 1009)
1	(1027, 862)
5	(1067, 1029)
11	(904, 1131)
28	(1322, 647)
33	(925, 1513)
49	(986, 458)
56	(410, 1585)
62	(1514, 998)
75	(638, 837)

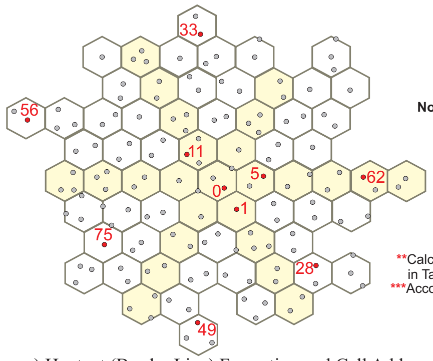
a) Network topology



Node ID	Location	Cell Name*
0	(1007, 1009)	(0, 0)
1	(1027, 862)	(1, -1)
5	(1067, 1029)	(2, 0)
11	(904, 1131)	(-1, 1)
28	(1322, 647)	(5, -3)
33	(925, 1513)	(-1, 5)
49	(986, 458)	(-1, -5)
56	(410, 1585)	(-10, 2)
62	(1514, 998)	(8, 0)
75	(638, 837)	(-6, -2)

*Calculated by Algorithm 2

b) Virtual Hexagonal Cell Formation



Node ID	Cell Name	Cell** Address	Border Node***
0	(0, 0)	(0, 0)	Yes
1	(1, -1)	(1, 5)	Yes
5	(2, 0)	(1, 0)	Yes
11	(-1, 1)	(1, 2)	Yes
28	(5, -3)	(4, 21)	No
33	(-1, 5)	(5, 8)	No
49	(-1, -5)	(5, 22)	No
56	(-10, 2)	(6, 16)	No
62	(8, 0)	(4, 0)	Yes
75	(-6, -2)	(4, 14)	No

**Calculated by transformation rules in Table 5.2
***According to Definition 1

c) Hextant (Border Line) Formation and Cell Addressing

Figure 5.5: *Honeycomb cell and rendezvous region formation of Honeycomb Architecture in the network setup phase*

Algorithm 3 Hexagonal Cell-based Data Dissemination

```
1:  Input: $[H, I]$, address of the current cell
2:  Input: $[H_s, I_s]$, address of the sink's current cell
3:  Output: $[H_o, I_o]$, address of next hop cell
4:  I. Find next hop cell towards center
5:      $k = \lceil I/H \rceil$
6:      $[H_o, I_o] \Leftarrow [H - 1, I - k]$
7:  II. Find next hop cell towards sink
8:  $k = \lceil I_s/H_s \rceil$
9:  $H_o \Leftarrow H + 1$
10: if $H_o <= kH_s - I_s$ then
11:     $I_o \Leftarrow I + k - 1$ // In the border line
12: else
13:     $I_o \Leftarrow I + k$ // within the hextant
14: end if
```

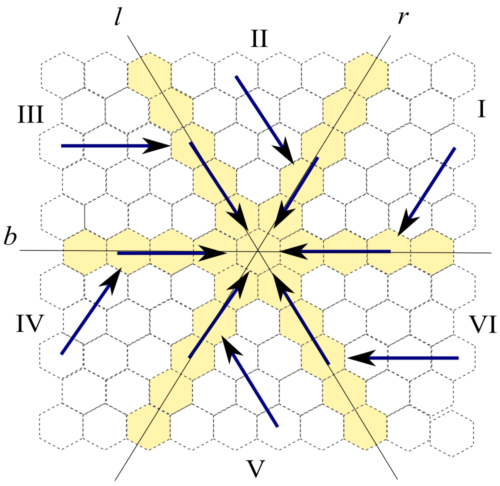
5.3.2 Hexagonal Cell-Based Data Dissemination

In the proposed data dissemination protocol, we use the concept of central re-dissemination in which the packets flow towards the center cells following previously selected directions. Instead of sending packets directly to the center cell by using a simple geographic routing, we send data through border lines towards the center cell. The aim is to store the generated data reports in the border lines so that the mobile sinks can easily collect them using a query-based data reporting method. However, our approach is purely geographical, which means that we do not use flooding for route setup. The only required information is the node position which is associated with a hexagonal cell in the honeycomb architecture. With the given virtual infrastructure, we propose Hexagonal cell-based Data Dissemination (HexDD) having the steps of (i) event data forwarding from sources, (ii) querying, and (iii) event data delivery to sinks, which are explained in the following.

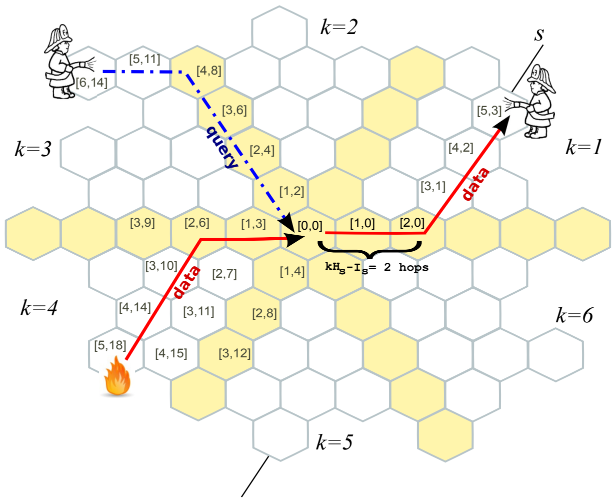
Event Data Forwarding

Event data forwarding in HexDD is done through border nodes towards center region according to *Algorithm 3-I* starting at line 4 of *Algorithm 3*. Line 5 of *Algorithm 3* calculates the hextant number k of the current cell of the node which has the data packet. Line 6, then, determines the next cell to forward the data packet. To find next hop, H of current cell is reduced by one because packet will be forwarded to the cell which is 1-hop closer to the center and I is reduced by k since the difference between Is of two adjacent cells on the packet forwarding direction (see Figure 5.6) of a hextant is equal to k for all hextants. As shown in Figure 5.6(a) with arrows, sensors route the packets to border cells in the first line segment of the hextant, e.g. line r for hextant II, following a direction parallel to the second border line of the hextant, e.g. line l for hextant II. When the data reaches one of the diagonal lines, it is forwarded along the border line towards the center cell.

Sensors in the border lines act as rendezvous points for data storage and look-up which means border nodes have a replica of data in their cache. When a sensor on the border line receives a new data packet from a source node, it updates its record with the new data so



(a) Packet Forwarding Directions



(b) Data and Query Dissemination

Figure 5.6: *HexDD in Honeycomb Tessellation*

it keeps the most up-to-date data packet. Another option can be logging all the data in the border nodes from the beginning of the event; however, this requires a lot of memory.

To facilitate the data lookup process, two replication schemes are possible in the border lines: the data can be either stored in all nodes of hexagonal cells or just in the cell-leader of each cell. The first scheme needs a fine-tuning of *border line width*, w, to prevent an increase of congestion under high traffic load conditions, while the second one requires a periodic cell-leader election and a replication mechanism. As in [75] and [151], we disregard the lines' width w. We assume that each border line covers only one cell (see Figure 5.6(a)).

The HexDD keeps the traffic flow in all regions of the network nearly balanced because honeycomb architecture divides the network space into six partitions and each partition uses a different border line segment for data dissemination; therefore, the traffic is spread among the different border lines.

Querying

In order to retrieve specific data, a sink sends a query towards the center by using *Algorithm 3-I*. The data and query packets are sent towards the center by using the same forwarding directions which are shown in Figure 5.6(a). The first border node which receives the query forwards it towards the center cell. Each node in the border cells checks its cache when it receives a query. If the data requested is in the cache of a border node, it sends data back to the sink. Replicating data on the border cells can decrease the cost of data look-up and the data delivery latency.

Event Data Delivery to Sink

To send data towards the sink, the reverse path of the sink's query forwarding path can be calculated by using the cell address of the sink as given in the *Algorithm 3-II* starting at line 7 of *Algorithm 3*, or can be stored in the query packet. The forwarding directions of the data packets from center to the sinks are exactly the opposite directions of the arrows shown in Figure 5.6(a). Line 8 of the *Algorithm 3* calculates the hextant number of the sink's current cell. Line 9 increases the H by one to get to the next hexagonal ring which is 1-hop closer to the hexagonal ring of the sink's cell. The data first travels in one of the border lines according to hextant number of sink's cell. In line 10, H is compared with $kH_s - I_s$ to determine the number of hops that the packet should be forwarded along the border line. Indeed, this check helps to find the turning point of the message towards the sink's cell. If the packet is still on the border line, I is increased by $k - 1$ in line 11. When the packet reaches the cell which is on the same line (i.e. line s parallel to line r in Figure 5.6(b)) where sink's cell is also located, the packet is forwarded towards the inside of the hextant. Within the hextant, I of the current cell is increased by k in line 13 until the packet reaches the cell of sink.

Before sending data to a regular node, the algorithm always checks if there is a sink node in the next hop cell. If so, the data is sent to the sink in the next cell. Otherwise, it sends the data packet to a sensor node in the next cell until the packet reaches to a sink.

Figure 5.6(b) shows the data and query dissemination in HexDD. If there is no neighbor node to forward the packet (i.e. query or data packet) in the next 2-hop cells calculated by the *Algorithm 3*, the protocol switches to route recovery procedure explained in the following section.

5.3.3 Fault tolerance

Algorithm 3 assumes that there is at least one node which will perform multi-hop routing within each cell. However, this may not be always the case. Sometimes an area of the network can be lost for different reasons, e.g., environmental reasons such as fire. Holes are created where there is a group of cells that do not have any active node inside. We propose a hole detection and bypassing mechanism, which is one of the most important features that shows how we maintain the honeycomb architecture even if a part of the network is lost.

A sensor can easily detect the hole region by checking its neighbor table, which is updated by periodic beacon packets. If the sensor has no neighbor on the next 2-hop cells in its radio

Algorithm 4 HexDD with Hole Recovery

```
1: Input: $[H, I]$, address of the current cell
2:     $[H_s, I_s]$, address of the sink's current cell
3:     $N = \{n_1, ..., n_m\}$, list of neighbors
4:     $N_a = \{[H_1, I_1], ..., [H_m, I_m]\}$, list of cell addresses of neighbors, where node $n_m$ is in cell $[H_m, I_m]$
5: Output: $n$, next hop neighbor to forward the packet
6: I. Find next hop neighbor towards center
7: $[H_c, I_c] \Leftarrow$ Find next hop cell towards center (Alg. 3-I)
8: if $[H_c, I_c] = [H_i, I_i] \in N_a$ then
9:     $n \Leftarrow n_i$ {forward data to neighbor in next cell}
10: else {there is a hole, enter route recovery}
11:     $n \Leftarrow n_j$ with $H_j$ the smallest $H$ in $N_a$
12: end if
13: II. Find next hop neighbor towards sink
14: $k = \lceil I_s/H_s \rceil$
15: $p = I_s - (k-1)H_s$
16: if $[H, I]$ in the regular path (Alg. 5) then
17:     $[H_c, I_c] \Leftarrow$ Find next hop cell towards sink (Alg. 3-II)
18:     if $[H_c, I_c] = [H_i, I_i] \in N_a$ then
19:         $n \Leftarrow n_i$ {forward data to neighbor in next cell}
20:     else {there is a hole, enter route recovery}
21:         $n \Leftarrow n_j$ with $[H_j, I_j]$ where $|H_s - H_j| + |I_j - (k-1)H_j - p|$ is the minimum in $N_a$
22:     end if
23: else {packet is already in the route recovery}
24:     $n \Leftarrow n_j$ with $[H_j, I_j]$ where $|H_s - H_j| + |I_j - (k-1)H_j - p|$ is the minimum in $N_a$
25: end if
```

range, it concludes that there is a hole at that area of the network. *Algorithm 4*[*] gives the details of HexDD with hole recovery.

Algorithm 4-I explains the route recovery when sending packets towards the center. Line 7 calculates the next hop cell using *Algorithm 3-I* and line 8 checks if there is a neighbor in the next cell. If there is no neighbor in the next cell, the algorithm enters route recovery in line 10. To find an alternative path, in line 11, the sensor sending its packet (i.e. data or query) towards center checks its neighbors and chooses the neighbor having the smallest H, which shows the shortest cell-count of the node from the origin cell (see node C in Figure 5.7).

Algorithm 4-II explains the route recovery when the data is being sent from the center to the sink. In line 15, p, the maximum number of hops between the cell of the sink and the first border line, is calculated. That is the number of hops between lines s and b (i.e. first border line of the hextant) in Figure 5.7. Line 16 checks if the current node is in the regular path of the packet according to *Algorithm 5* to know if the packet is already in the route recovery or not. If the packet is in the regular path, in line 17, the next hop cell is calculated

[*]For simplicity in Algorithm 4 we show neighbor checking for only next 1-hop cell.

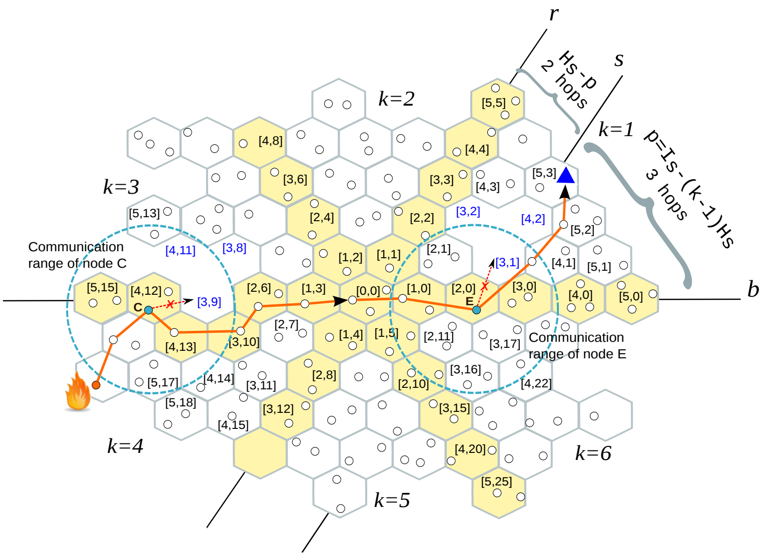


Figure 5.7: *Fault tolerance mechanism in HexDD*

Algorithm 5 Regular Path Check

```
1: **Input**: $[H, I]$, address of the current cell
2:     $[H_s, I_s]$, address of the sink's current cell
3:     $p$, the maximum number of hops between the cell of the sink and the first border line
4: **Output**: *true* (i.e. in the regular path) or $false$ (i.e. off the regular path)
5: if $[H <= H_s - p \wedge I = Hk - (H_s - p)] \vee [H > H_s - p \wedge I = (H_s - H)k]$ then
6:     return $true$;
7: else
8:     return $false$;
9: end if
```

based on *Algorithm 3-II*. If there is no neighbor in the next hop cell, the packet enters route recovery at line 20. In line 21, the packet is forwarded to the neighbor n_j within cell $[H_j, I_j]$ where H_j is the closest to H_s and I_j is the closest to $p + (k - 1)H_j$ in neighbor list, N_a. This approach achieves to forward the data packet to the cell which is on the hexagonal ring that is the closest to the hexagonal ring of the sink. At the same time, it tries to keep the same distance from the second border line (i.e. line r) as sink. In Figure 5.7, where both the sink and the node E are located on the line s, node E in cell $[2, 0]$ forwards the packet to the cell $[4, 1]$ according to the rule in line 21. If packet is already in the route recovery, it applies the same rule in line 24.

This mechanism is simple and efficient since it avoids to flood any other control message to inform other nodes about the hole, which is required to find new bridge nodes. This is mainly the advantage of using honeycomb tessellation. It is important to point out that in HexDD, if a hole happens at the center of the network, the crossing area of the border lines

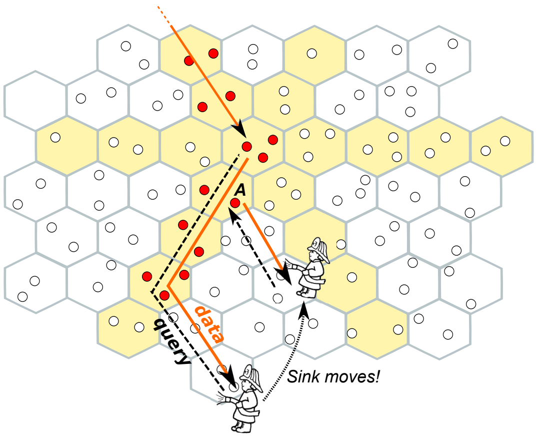


Figure 5.8: *Mobility of sink in HexDD*

at the central region should be shifted to a closer location which is not affected by the hole, or the first possible hexagonal ring which excludes the hole can become the central region.

Instead of calculating forwarding path between the center and a sink by *Algorithm 4-II*, the reverse path can be stored in the query. Since the sink sends a new query whenever it changes its cell, saving the path in the query is also efficient. The reverse path in the query recovers the hole at the path back to sink (i.e. assuming communication links are bidirectional) because when the query is being sent towards the center, the alternative path is calculated and stored in the query. However, if a new hole is formed on the path back to the sink, the reverse path stored in the query packet will not be valid anymore.

5.3.4 Resiliency to localization errors

The hexagonal tessellation and geographic forwarding protocol rely on each node being able to estimate its own coordinates. These estimates are highly likely affected by a non-negligible error, which in turn affects the calculated cell addressing $[H, I]$ used for packet forwarding. We use a kind of polar coordinate system to address the cells of the tessellation. This addressing scheme serves as a positioning (coordinate) system that is rougher than the coordinates of the sensor nodes, with a precision appropriate for the transition range. A localization estimate with an error $err < r$, where r is the edge length of a hexagonal cell, will result in the same cell address $[H, I]$. Therefore, the packet forwarding mechanism will not be affected by the localization errors. If a given node, which is close to the boundary of its hexagonal cell, calculates a wrong cell address due to localization error, the erroneous cell address will be one of the neighbor cells of its real cell. If there is no other node in the real cell, the fault-tolerance mechanism can easily find another cell to forward the packets.

5.3.5 Mobility Support

The mobility of WSN, where most of the sensors are stationary, can be divided into the *stimulus mobility* and *sink mobility*. The impact of stimulus mobility on the dissemination scheme is very small because when stimulus moves to another cell, a sensor that captures the stimulus sends the data towards the center. Also, if the sink moves inside its current cell, there is no need for another process since the data will be forwarded to the same neighboring cell until the sink leaves its cell. When the sink moves to another cell, it needs to send a new query message towards the center to inform the center nodes about its new cell. If any border node has the requested data in its cache (see node A in Figure 5.8), it directly sends data to the new cell of the sink.

Although it is assumed that sensor nodes are stationary in our work, HexDD can also handle mobility of sensor nodes. Sensors can easily recalculate their new hexagonal cells by using Algorithm 2 while they are moving. However, the uniform deployment of the sensor network should not be affected by the mobility of sensors. Thus, HexDD allows for a limited mobility of sensor nodes, meaning the number of mobile nodes is low in a high density network so that the risk of changing the uniform distribution of network coverage is low.

5.4 Performance Analysis

This section provides an analytical study of communication cost and hotspot traffic cost of HexDD and other protocols given in Section 5.2. The *communication cost* represent the total amount of messages generated in the network during the data dissemination and look up process. It is important to estimate communication cost since it has a direct influence on the network lifetime. The *hotspot traffic cost* is the total energy consumption of one single node located at hot regions. It is also important because it restricts the network scalability and lifetime.

5.4.1 Analysis Model and Assumptions

We consider a network with large number of nodes being deployed uniformly and distributed over a unit area. We use the function $H(l)$ as the number of hops on a path between two arbitrary nodes x and y such that $|x, y| = l$ is the euclidean distance between these two nodes. According to [54], given a geographical routing protocol, we have $H(l) = \zeta \frac{l}{r}$ where r is the communication range and $\zeta \geq 1$ is a scaling factor which depends on the spatial node density λ. For simplicity in our analytical analysis, we assume that $\zeta = 1$.

For conformity with the analysis in [150], we consider four types of messages: event notification, query, data, and control messages, whose sizes are p_e, p_q, p_d, and p_c respectively. We consider m sinks moving randomly in the sensor field as well as n sources. Each sink generates a number of queries equal to $\bar{q}$ and each source generates a number of events equal to $\bar{e}$. Thus, the total number of queries and events can be written as $m\bar{q}$ and $n\bar{e}$.

5.4.2 Communication Cost

The *total communication cost* is the sum of the communication cost brought by all control messages, event notification messages, queries, and data messages. In other words, it represents the total number of messages generated in the network during the data reporting, data lookup and data collection processes. The total communication cost is the summation of three components:

(i) C_{DD}: cost of data reporting to the rendezvous region

(ii) C_{DL}: cost of data lookup (query dissemination) to the rendezvous region

(iii) C_{DT}: cost of transferring data from the rendezvous region to a sink

Therefore, the total communication cost of a given protocol is $C_{protocol} = C_{DD} + C_{DL} + C_{DT}$. We use the following metrics in the calculations: (i) $D_{src,rdv}$ – the distance between the source node and the rendezvous region, (ii) $D_{sink,rdv}$ – the distance between sink and the rendezvous region, (iii) $D_{rdv,sink}$ – the distance between the rendezvous region and sink. In what follows, we compare the HexDD, LBDD, TTDD, GHT and RailRoad protocols. Figure 5.9 shows the worst case scenarios for each protocol, which is considered in the calculations.

HexDD: In case of HexDD, upon the detection of a new event, the sensor node sends the sensor reading towards one of the border lines and then to the central region. We assume that there is at least one node in each cell. In the worst case, this message travels the path from source to rendezvous point (RP), $D_{src,rdv} = \frac{1}{2} + \frac{\sqrt{3}}{6} \approx 0.79$, in Figure 5.9(a) and meets about $H(0.79)$ nodes. To retrieve data, a mobile sink sends a query message which is forwarded towards one of the border lines and then forwarded to the center. In worst case, the query travels the path from sink to RP, $D_{sink,rdv} = \frac{1}{2} + \frac{\sqrt{3}}{6} \approx 0.79$, in Figure 5.9(a) and meets about $H(0.79)$ nodes. After query and data meet at the central region, data packet is transferred from the RP to the sink, $D_{rdv,sink} = \frac{1}{2} + \frac{\sqrt{3}}{6} \approx 0.79$, and meets in the worst-case $H(0.79)$ nodes. Therefore, the total communication cost of HexDD is

$$C_{HexDD} = n\bar{e}p_d H(0.79) + m\bar{q}p_q H(0.79) + n\bar{e}p_d H(0.79).$$

LBDD: In the case of LBDD, upon the detection of a new event, the sensor node sends the measured data towards the line. In the worst case, this message travels $D_{src,rdv} = 0.5$ in Figure 5.9(b) and meets about $H(0.5)$ nodes. To retrieve the data, a mobile sink sends a query message which is forwarded greedily towards the line. This message is then propagated along the line until it is received by the corresponding inline-node. In the worst case, the query travels $D_{sink,rdv} = 0.5 + 1$ and meets about $H(1.5)$ nodes. Then, the data is transferred from the inline-node to the sink, traveling $D_{rdv,sink} = \sqrt{5}/2 \approx 1.12$ and meets in the worst-case $H(1.12)$ nodes (diagonal of a half square). To avoid the transfer of duplicated data, we suppose that a sink receives a response to its query only if the inline-node owns a new data. The total communication cost of LBDD in the worst case is then

$$C_{LBDD} = n\bar{e}p_d H(0.5) + m\bar{q}p_q H(1.5) + n\bar{e}p_d H(1.12).$$

GHT, TTDD, and RailRoad: The total communication cost of GHT, TTDD, and RailRoad are calculated in a similar way. As show in Figure 5.9(c), for GHT, $D_{src,rdv} = D_{sink,rdv} = D_{rdv,sink} = \sqrt{2} \approx 1.41$. For TTDD calculation, each term indicates the communication costs of grid construction, query forwarding, and data forwarding, respectively. As show in Figure 5.9(e), in RailRoad, $D_{src,rdv} = D_{sink,rdv} = \sqrt{2}/4 \approx 0.35$, and the perimeter of the Rail is 2. Each term indicates the communication cost of event notification, query forwarding, query circulation, query notification and data dissemination (for further details, refer to [150]).

$$C_{GHT} = n\bar{e}p_d H(1.41) + m\bar{q}p_q H(1.41) + n\bar{e}p_d H(1.41);$$

$$C_{TTDD} = n\frac{4\lambda}{H(\frac{1}{c})}p_c + m\bar{q}[\lambda c^2 + H(2)]p_q + n\bar{e}[H(2) + H(\sqrt{2}/(2c))]p_d;$$

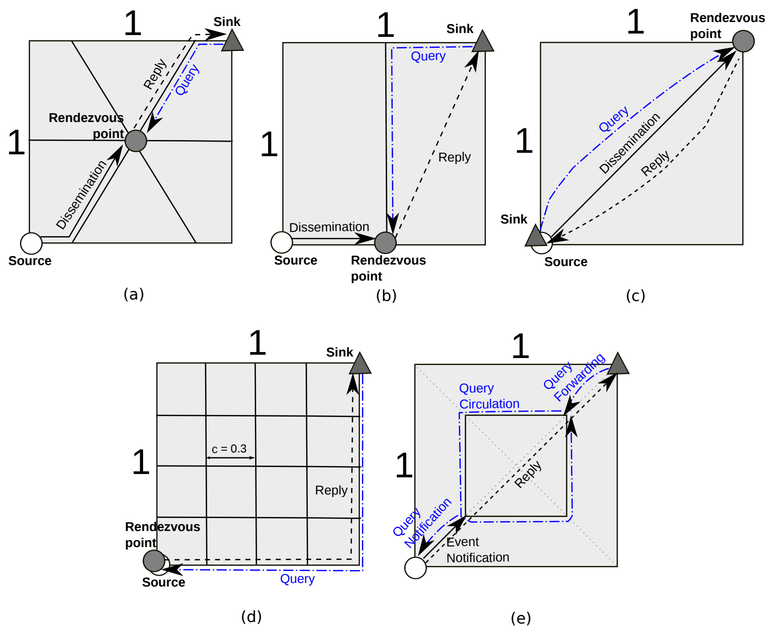
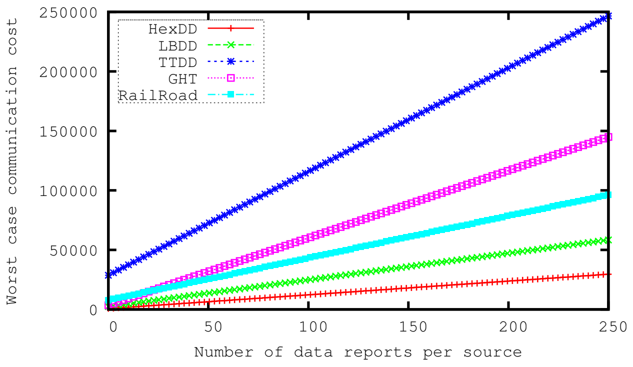


Figure 5.9: *Worst case scenarios for (a) HexDD: Hexagonal cell-based Data Dissemination, (b) LBDD: Line-based Data Dissemination, (c) GHT: Geographic Hash Table, (d) TTDD: Two-tier Data Dissemination, (e) RailRoad*

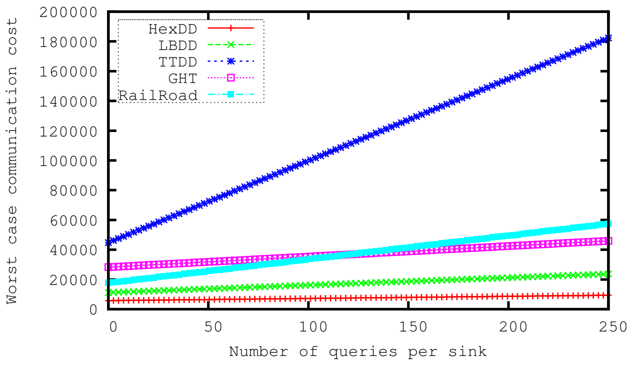
$$C_{RailRoad} = n\bar{e}p_e H(0.35) + m\bar{q}p_q H(0.35) + m\bar{q}p_q H(2\sqrt{2}) + n\bar{e}p_q H(0.35) + n\bar{e}p_d H(\sqrt{2}).$$

Figure 5.10 shows the comparison of the worst-case communication costs of all approaches for two scenarios. We consider 1000 sensor nodes deployed on a square sensor field of size 1×1. The sensor coverage area radius is $r = 0.1$ and we suppose that $c = 0.25$ (the size of a TTDD cell). The first scenario considers a fixed number of queries per sink ($\bar{q} = 50$) with a varying number of data reports per source node. The results for the first scenario are shown in Figure 5.10(a). In the second scenario, we consider a fixed number of data reports per source ($\bar{e} = 50$) for a varying number of queries per sink. Results for the second scenario are shown in Figure 5.10(b).

We notice that TTDD presents a rather high communication cost in both scenarios resulting from its need to build grids and its routing strategy along the grid. RailRoad and LBDD, which implement a large virtual infrastructure, are more suitable for scenarios with a high number of data reports as show in Figure 5.10(a). The reason is that the infrastructure reduces the communication path and thus the cost between the source and the node having the disseminated data. On the other hand, the protocols GHT and LBDD are more suitable for scenarios with a large number of queries because these protocols propose a low look-up cost as shown in Figure 5.10(b). Finally, HexDD, which combines a large infrastructure with a central re-dissemination strategy reducing the data look-up cost, presents a lower commu-



(a) $\bar{q} = 50$



(b) $\bar{e} = 50$

Figure 5.10: *Worst case communication costs (m = 5 sinks, n = 10 sources)*

nication cost in both scenarios.

5.4.3 Hot Region Traffic Cost

In rendezvous-based protocols it is important to estimate how densely messages are concentrated on the rendezvous area. *Hot region traffic cost* is the average energy spent by a hotspot region node. In data-centric storage such as GHT, all messages are directed to several home nodes. To prevent home nodes from being exhausted due to heavy traffic, replicas of home nodes are chosen. This approach, however, increases the total energy consumption and the replication cost of home nodes. In Railroad, every query and event summary is sent to the Rail, which can be the bottleneck that limits the network lifetime. Also, in HexDD queries and data packets are forwarded toward border lines, which are becoming hot regions in the network. In this section we analyze the hot region traffic cost of GHT, RailRoad, LBDD and our approach HexDD. In the following calculations, T is the amount of energy for a node to transmit a single bit, and R is the energy needed to receive a bit.

In data-centric storage, the home nodes can be hot spots, and the hotspot traffic cost can be written as follows [151]:

$$EH_{GHT} = R[\frac{n\bar{e}}{s}p_d + \frac{1}{\gamma}\frac{m\bar{q}}{s}p_q] + T[\frac{1}{\gamma}\frac{n\bar{e}}{s}p_d]$$

where γ is the number of nodes in a replica set including the home node. It means a home node has $\gamma - 1$ replicas. When γ is set to 1, there exists no replica nodes but the home nodes. There are s different event types in the network. We assume that there is only one event type ($s = 1$) in the network for the calculations. All data and query packets coming from sources and sinks are received by the home node (i.e. $R(n\bar{e}p_d + \frac{1}{\gamma}m\bar{q}p_q)$). The home node, then, transmits the data packet to the sinks (i.e. $T(\frac{1}{\gamma}n\bar{e}p_d)$).

The hotspot traffic cost of Railroad can be written as follows [151]:

$$EH_{RR} = \frac{1}{N_R}\left[Rn\bar{e}p_eN_{ST} + Rm\bar{q}p_qN_{RT} + Tn\bar{e}p_e + Tn\bar{e}p_q + Tm\bar{q}p_qN_{RT}\right]$$

where N_R, N_{ST}, and N_{RT} stand for the number of the rail nodes, the number of rail nodes in a station[†] , and the number of nodes that a query stays in a single tour around virtual *rail*, respectively. In the event notification process, one node out of N_{ST} nodes transmits the event notification packet (i.e. $Tn\bar{e}p_e$) sent by a source node and N_{ST} nodes receive this event notification packet (i.e. $Rn\bar{e}p_eN_{ST}$). For query flooding in the *rail*, N_{RT} nodes out of N_R nodes receive a query packet (i.e. $Rm\bar{q}p_qN_{RT}$) sent by a sink and N_{RT} nodes out of N_R nodes transmit the query packet (i.e. $Tm\bar{q}p_qN_{RT}$). Finally, one node out of N_{RT} nodes transmits the query packet to the source node (i.e. $Tn\bar{e}p_q$). The data is directly sent from source to sink with greedy forwarding.

The hotspot traffic cost of LBDD can be written as follows:

$$EH_{LBDD} = \frac{1}{N_L}\left[Rn\bar{e}p_dN_{ST} + Rm\bar{q}p_qN_L + Tn\bar{e}p_d + Tm\bar{q}p_qN_L\right]$$

where N_L is number of inline nodes and N_{ST} is the number of inline nodes in a station which is a small group of nodes in the virtual line. In the data dissemination process, N_{ST} nodes out of N_L nodes receive a data packet (i.e. $Rn\bar{e}p_dN_{ST}$) sent by a source node. For query flooding in the line/strip, N_L nodes receive the query packet (i.e. $Rm\bar{q}p_qN_L$) sent by a sink and N_L nodes transmit the query packet (i.e. $Tm\bar{q}p_qN_L$). Finally, one node out of N_{ST} nodes sends the data packet to the sink (i.e. $Tn\bar{e}p_d$). Greedy forwarding is used to send data to the sink.

The hotspot traffic cost of HexDD is as follows,

$$EH_{HexDD} = \frac{1}{3N_{BL}}\left[2Rn\bar{e}p_d\frac{N_{BL}}{2N_C} + Rm\bar{q}p_q\frac{N_{BL}}{2N_C} + 2Tn\bar{e}p_d\frac{N_{BL}}{2N_C} + Tm\bar{q}p_q\frac{N_{BL}}{2N_C}\right]$$

where N_{BL} is the number of border nodes in a diagonal line, and N_C is the average number of nodes in a cell. $N_{BL}/2N_C$ is the number of cells on a border line[‡]. Since one node per cell transmit or receive the packets, it is the number of nodes having the packets on a border line.

[†]In RailRoad and LBDD, the rendezvous region is divided into smaller subregions called *station*. All the nodes in a station are informed about the data but one of them forwards data towards sink.

[‡]Border line is the half of a diagonal line.

In data dissemination and data transfer process, a node in each cell receives and transmits the data packet along the diagonal line (i.e. $2Rn\bar{e}p_d\frac{N_{BL}}{2N_C} + 2Tn\bar{e}p_d\frac{N_{BL}}{2N_C}$). The sink's query travels the border line (i.e. $Rm\bar{q}p_q\frac{N_{BL}}{2N_C} + Tm\bar{q}p_q\frac{N_{BL}}{2N_C}$). The simplified version of the hotspot traffic cost of HexDD is:

$$EH_{HexDD} = \frac{1}{3N_C}\Big[Rn\bar{e}p_d \;+\; (0.5)Rm\bar{q}p_q \;+T n\bar{e}p_d \;+\; (0.5)Tm\bar{q}p_q\Big]$$

The EH_{HexDD} calculated above is the average energy spent by a border node. However, the most critical region of the hexagonal architecture is the center cell. The hotspot traffic cost of HexDD at the center cell is:

$$EH_{HexDD_C} = \frac{1}{N_C}\Big[Rn\bar{e}p_d \;+\; Rm\bar{q}p_q \;+T n\bar{e}p_d\Big]$$

The nodes at the center cell receive all events (i.e. $Rn\bar{e}p_d$) and all queries (i.e. $Rm\bar{q}p_q$), also transmit all events to the sinks (i.e. $Tn\bar{e}p_d$). When we compare EH_{HexDD} and EH_{HexDD_C}, we observe that a node at the center cell processes around 6 times more traffic[§] than a node on a border line. This is an expected result since each border line processes the traffic which comes from one of the six hextants while the center is processing the traffic coming from all hextants.

To calculate the hotspot traffic costs of the protocols, the number of sources n and number of sinks m vary between 1 to 18 in the first set of analysis. In the second set of analysis, we set n to 5 and m to 15. Total number of nodes N in a sensor field of $1000m \times 1000m$ is set to 10000. N_R is 8% of total nodes and N_{RT} is 480. In the analysis, we use the same values used in [151] for N_{ST}, and R/T which are taken as 16 and 3/8, respectively. Both the width of the Rail and the station are set to 40 m, that is the radio range of the sensor nodes. Hence, the average number of nodes in a cell, N_C, is 3. The width of the rendezvous region of LBDD and RailRoad affects the number of rendezvous nodes. Larger rendezvous area results in higher energy consumption inside the rendezvous area and also higher hotspot traffic costs. We set $p_e = p_c = p_q$ and $p_d = 2 \times p_q$.

In Figure 5.11, 5.12, and 5.13 we show the hotspot traffic cost of HexDD compared with other protocols. In the first graphs of the figures x axes are the number of sinks (m) and y axes are the number of sources (n). In the second graphs of the figures x axes are the total number of queries ($m\bar{q}$) and y axes are the total number of events ($n\bar{e}$). Finally, the z axes are the ratio between the hotspot traffic cost of HexDD (EH_{HexDD}) and the hotspot traffic cost of another protocol ($EH_{protocol}$). A border node in HexDD processes less data than a rendezvous node in the other protocol if the ratio $EH_{HexDD}/EH_{protocol} < 1.0$. In the first set of graphs, the aim is to see the effect of varying number of sinks and sources on the hotspot traffic costs of the protocols. The second set of graphs shows the hotspot traffic costs in the *event-driven* scenario, where the number of event messages per source ($\bar{e}$) is larger than the number of queries per sink ($\bar{q}$), and in the *query-based* scenario, where the number of queries per sink is larger than the number of event messages per source.

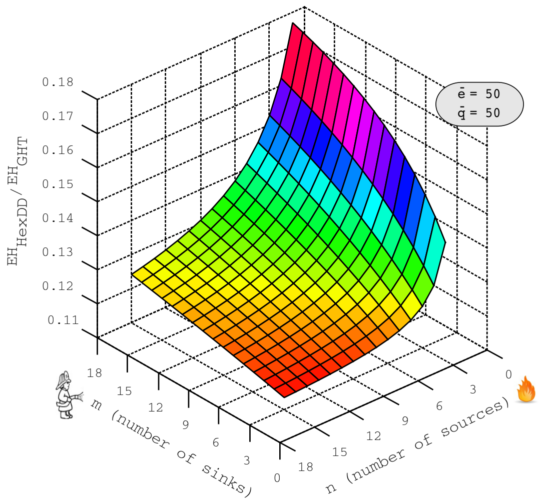
Figure 5.11(a) shows the hot region traffic cost of HexDD compared with the data-centric storage GHT with varying number of sinks and sources. The result shows that a home node

[§]Indeed, it is less than 6 times since center nodes do not transmit queries.

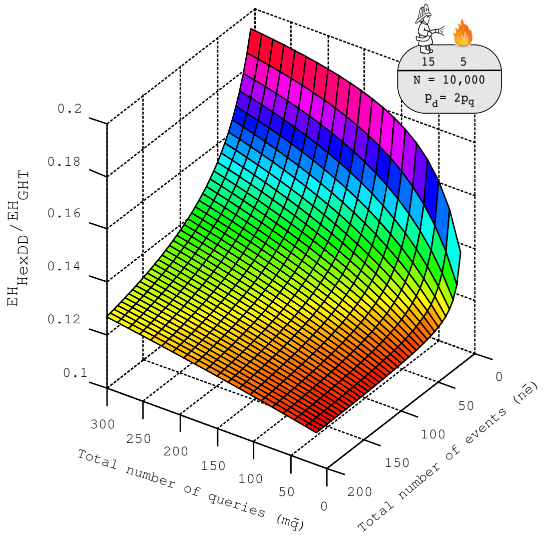
in a data-centric storage has to process much more requests than a border node in HexDD protocol. This is more remarkable as the number of sources increases and the number of sinks decreases. In Figure 5.11(b), where we vary the number of queries per sink and the number of data reports per source, the same behavior is observed as the total number of events increases and the total number of queries decreases. Both graphs show that the hotspot traffic cost of HexDD is much less than that of a data-centric storage.

In Figure 5.12 we compare the hot region traffic costs of HexDD and RailRoad. The results in Figure 5.12(a) show that when we have many event sources but a couple of sinks in the network (i.e. see $n = 15$, $m = 3$, and $EH_{HexDD}/EH_{RR} = 1.8$ in the figure), a border node in HexDD processes much more requests than a rail node in RailRoad. This is due to the fact that RailRoad does not process/forward data reports in the Rail region; on the other hand, in HexDD diagonal lines are also used for data forwarding to cache data on the border nodes for sink queries. This is an expected result because HexDD is designed for networks where the difference between the number of sinks and sources is not very high. For instance, when $n = 15$ and $m = 6$, the ratio $EH_{HexDD}/EH_{RR} = 0.98$ so HexDD is still better than RailRoad. As observed in the figure, when the number of sinks is greater than or equal to the number of sources, the hotspot traffic cost of HexDD is much less than that of RailRoad. Figure 5.12(b) presents the results of a scenario having 15 sinks and 5 sources in the network. Apparently, HexDD becomes advantageous over RailRoad in terms of hotspot traffic cost in the *query-driven* scenarios, where the query generation rate is higher than the event generation rate. Also, when the total number of queries is close to the total number of events, HexDD still processes less requests on the rendezvous lines than RailRoad.

Figure 5.13, where the hot region traffic costs of HexDD and LBDD are compared, has a similar behavior of the previous graphs for RailRoad comparison because the ratio $EH_{RR}/EH_{LBDD} \simeq 0.6$ for the given network. That means an inline node of LBDD already processes more requests in the line-based rendezvous region than a rail node in RailRoad. Also, as shown in Figure 5.13(a) and (b), an inline node of LBDD processes much more requests than a border node of HexDD in most of the cases. The same observations previously discussed for RailRoad comparison are also valid for LBDD comparison. Only the ratio EH_{HexDD}/EH_{LBDD} is smaller than the ratio EH_{HexDD}/EH_{RR} for the same inputs. For instance, when $n = 15$ and $m = 6$, the ratio $EH_{HexDD}/EH_{LBDD} = 0.59$.

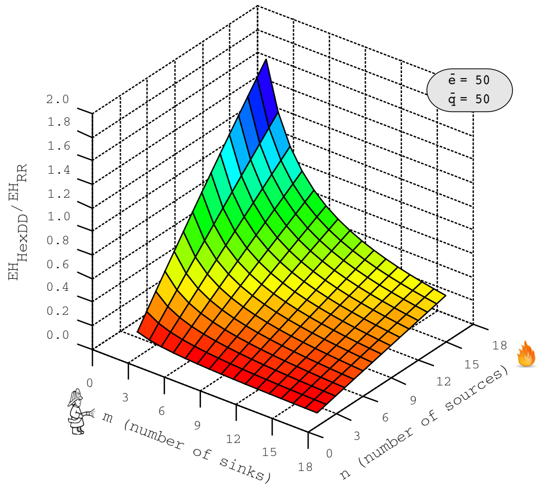


(a) Number of sinks vs. Number of sources

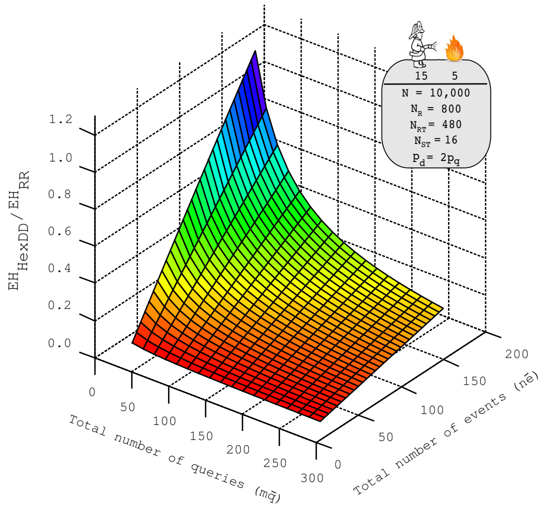


(b) Total number of queries vs. Total number of events

Figure 5.11: *Analysis of hotspot traffic cost of HexDD compared with GHT*

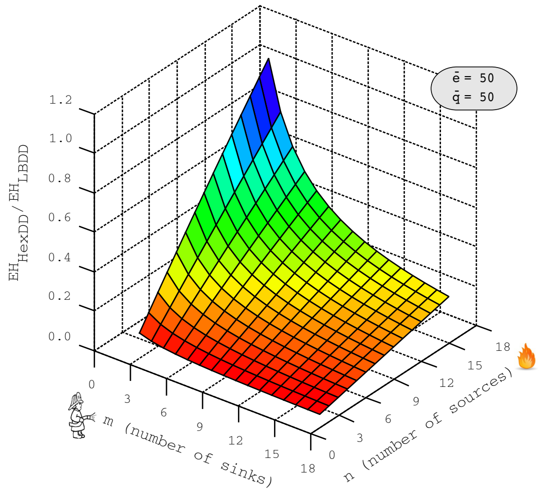


(a) Number of sinks vs. Number of sources

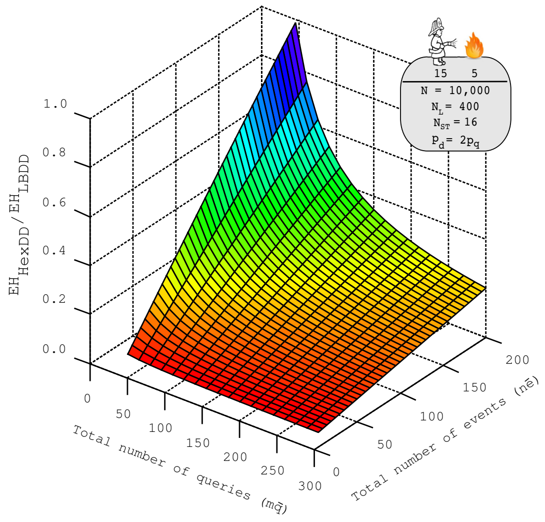


(b) Total number of queries vs. Total number of events

Figure 5.12: *Analysis of hotspot traffic cost of HexDD compared with RailRoad*



(a) Number of sinks vs. Number of sources



(b) Total number of queries vs. Total number of events

Figure 5.13: *Analysis of hotspot traffic cost of HexDD compared with LBDD*

5.5 Conclusions

In this chapter, our goal was designing a routing protocol that supports mobility of sink and source by keeping the data delivery ratio as high as possible and energy consumption and delivery delay as low as possible. We proposed a virtual infrastructure called *honeycomb tessellation* which allows an efficient geographical routing of event messages, namely *HexDD*. The HexDD uses the concept of rendezvous region for events and queries. The *highways*, which lie on three main directions of the network, are used as rendezvous areas. They make it faster for sinks to access data. Honeycomb tessellation offers advantages in terms of algorithmic simplicity in routing and fault tolerance against node failures. The HexDD protocol utilizes these properties of honeycomb tessellation for quick routing and hole recovery, assuring high data delivery ratio and low latency.

We have analytically evaluated the communication cost and hot region traffic cost of HexDD, comparing it with other protocols. We conclude that HexDD is a very suitable protocol for applications, where we have many mobile sinks and sources in the network. In the next chapter, we investigate the performance of HexDD in two different scenarios: (1) Emergency Response which incorporates a *moderate mobility*, and (2) Vehicular Sensor Networks which incorporate *high mobility*.

CHAPTER VI [*]

Data Dissemination in Different Applications of Mobile Multi-sink WSN

In this chapter we focus on the performance evaluation of Hexagonal Cell-based Data Dissemination (HexDD) protocol described in Chapter 5 in different applications. We consider two different classes of mobile wireless sensor networks with mobile sinks: mostly static and highly mobile networks. To be more specific, these applications are defined as follows: (i) Emergency Response Application – mostly static, which contains scenarios in which most of the sensors are static and some sensors are attached to people or vehicles such as firefighters or unmanned aerial vehicles moving at low or medium velocities, and (ii) Vehicular Sensor Network Application – highly mobile, which contains scenarios in which many sensors are attached to devices that move at high velocities such as cars. We evaluate and compare the performance of the HexDD protocol with other application-specific data dissemination protocols in terms of data delivery ratio, latency and energy efficiency. In addition, we analyze the effect of the hotspot zones on the performance of various protocols. Through extensive simulations, we also investigate the efficiency of our fault tolerance mechanism, which is proposed as a supporting method to recover from routing holes in the network.

[*]Some parts of this chapter appear in the book chapter with the title "Infrastructure Assisted Data Dissemination for Vehicular Sensor Networks in Metropolitan Areas" In Roadside Networks for Vehicular Communications: Architectures, Applications and Test Fields, IGI-Global, Submitted in April 2011 [8] and some parts of the chapter also is submitted in the journal paper [6].

6.1 Wireless Sensor Networks in Emergency Response

We described the emergency response application in Chapter 2.4. In this section, we further discuss the characteristics of a wireless sensor network designed for an emergency response scenario and evaluate the performance of the HexDD protocol in this scenario.

6.1.1 Motivating Scenario

The sink mobility assumption may be useful for numerous applications. A typical application is *emergency response*. As shown in Figure 2.3, sensors are randomly deployed by UAVs to monitor the area of interest, e.g., a forest in a fire fighting scenario, and detect dangerous events, e.g., fire in forest. Detection of such events is realized by event-detection algorithms, e.g. [34]. Sensors report an alarm (including data about the current situation of the event) to mobile sinks. Mobile sinks monitor the progression of the event and take the appropriate actions (e.g., sending location of the fire to the mission coordinators via a satellite). Therefore, the sink represents an important component of a wireless sensor network as it acts as a gateway between the sensor network and the end-users.

The sink mobility assumption can be enforced by the nature of the employed application. For example, in the fire fighting scenario, the mobile entities (e.g., firefighters, firetrucks, UAVs, etc.) of the network have primary tasks, whereas data collection is a secondary task. For example, firefighters fight cooperatively to eliminate fire in the field while UAVs are responsible for transporting load (e.g. water) near the fire field or deploy sensors to inaccessible areas of the network. Their mobility is regulated according to their primary tasks. In the meanwhile, they are informed by the source nodes about the current situation of the event as they carry sink nodes onboard. The firefighters are warned about the dangerous situation around them in time, for example, the spread of the fire, i.e., where it is spreading and how quickly. Therefore, from data collection point of view, the sink mobility is uncontrollable. Sinks move randomly around the network and get data from the sources. Moreover, in emergency response scenarios, the use of mobile objects for data collection makes harder the damage of such component. Indeed, if a static sink is located in the area of interest, it can be damaged by the dangerous event such as fire, thus making the sensors disconnected from the end-users. The mobile sinks enable a more reliable data collection in the dangerous/inaccessible regions.

6.1.2 Performance Evaluation in Emergency Response Applications

For the purpose of performance evaluation, we have compared the proposed protocol HexDD with two other rendezvous-based approaches, LBDD and TTDD. We choose TTDD and LBDD for the comparison since we would like to investigate the effect of using hexagonal tessellation instead of rectangular grids and using three diagonal lines acting as rendezvous area instead of only one line-based region. The simulations have been carried out to evaluate routing performance and the fault-tolerance performance of the protocols. We also investigate the effect of central region size on the HexDD protocol.

For this purpose, firstly, we analyze the protocols with varying number of sink-source pairs. Secondly, we explore the impact of sink mobility (i.e. sink's maximum speed) on the performance of these protocols. We then analyze the fault-tolerance performance and vary the total number of holes and the size of holes in the network. We also analyze the protocols' energy distribution maps which are important to see hotspot regions created by each protocol in the network. Indeed, since all data reports and queries are concentrated over the central region in HexDD protocol, the hotspot problem can arise, limiting the network lifetime and the scalability. To prevent the central cell from being a bottleneck, it is possible to design a larger central region, including the first hexagonal ring, to better distribute the load among its nodes. Therefore, we, finally, test the routing performance of HexDD with different size of central region (i.e. center with one cell or center with the first hexagonal ring) to see if we can reduce energy consumption per node at the center with a larger central rendezvous region.

Simulation Environment

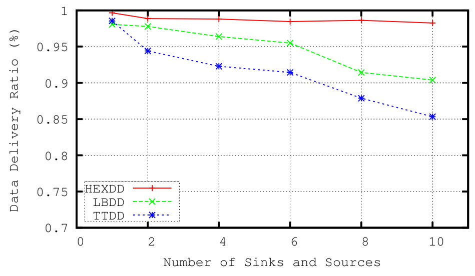
HexDD protocol is implemented and tested in NS2. To guarantee a fair comparison between TTDD and HexDD, we set simulation parameters comparable to those used in [173]. This includes simulation of IEEE 802.11 DCF as the underlying MAC and an energy model in which a sensor's transmitting, receiving and idling power consumptions are set to 0.66W, 0.395W and 0.035W, respectively. Although IEEE 802.15.4 is a standard developed to meet the needs for low-power and low-cost wireless communication, we prefer to use the IEEE 802.11 standard in our simulations to be comparable with previous works. A comparative performance study of IEEE 802.15.4 and IEEE 802.11 can be found in [178]. The cell size in TTDD is set to 600m. In LBDD, width of the virtual line is set to 250m. Each node has a transmission range of 250m. For routing performance simulations, 250 sensor nodes are randomly distributed on a 2000×2000 m^2 field. For fault-tolerance performance simulations, 210 nodes are randomly distributed on a 1500×1500 m^2 field. Each simulation run lasts for 200 seconds. Results are averaged over six random network topologies. A source generates one data packet per second, so there are in total 200 data packets/source sent. The sinks' mobility follows the standard *Linear Mobility* model. Mobile sinks could attain a maximum speed up to 14 m/s with 5 seconds pause time. The stimuli remain static during the simulation time. For different sets of simulations, speed and pause times of sinks are varied.

We use the following metrics to evaluate the performance of the protocols: (i) *Data Delivery Ratio:* defined as the ratio between the total number of data packets received by the sinks and the total number of data generated by the sources; (ii) *Data Delivery Delay:* defined as the total time elapsed between the data generation by a source and its reception by a sink, also averaged over all source-sink pairs; and (iii) *Energy Consumption:* defined as the communication (transmitting and receiving) energy the network consumes; the idle energy is not counted since it depends largely on the data generation interval and does not indicate the efficiency of the data dissemination protocol.

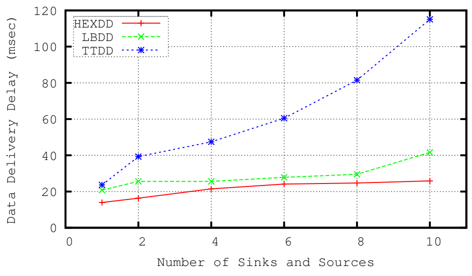
Simulation Results

Impact of the number of sink-source pairs: For the first set of simulations, the number of sink-source pairs is varied. Mobile sinks could attain a maximum speed up to 10 m/s with 5 seconds pause time. The stimuli remain static during the simulation time.

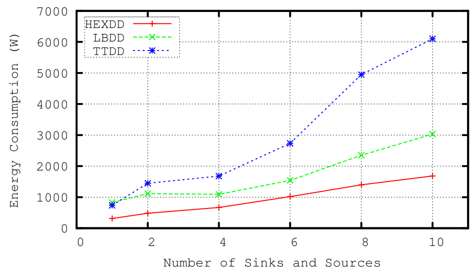
Figure 6.1(a) shows the data delivery ratio for the three protocols. We observe that the success rate slightly decreases as the number of sink-source pairs increases because of the congestion in the network. Although the results of HexDD, LBDD and TTDD are close, HexDD has the highest delivery ratio benefiting from the use of a virtual infrastructure of three border lines, which allow to better distribute the load among the nodes inside the rendezvous area. The delivery ratio of TTDD scheme falls more consistently as the number of



(a) Data Delivery Ratio (%)



(b) Data Delivery Delay (msec)



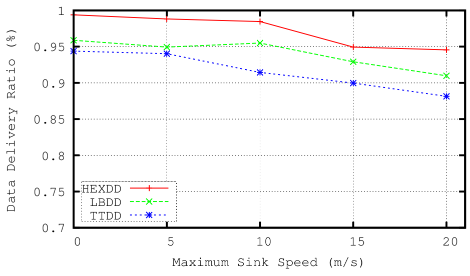
(c) Energy Consumption (W)

Figure 6.1: *Impact of the number of sink-source pairs*

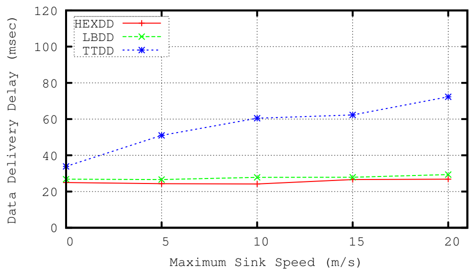
sink-source pairs grows.

Figure 6.1(b) presents the data delivery delay in seconds. We notice that in all protocols, the delay increases with the increase in number of sink-source pairs. However, the increase in the delay of TTDD is very large due to the time spent for the creation and the propagation of the grid for each source. Also, more sources generate more data packets, and more sinks need more local query flooding. Both increase the traffic volume and lead to longer delivery time. In HexDD, on the other hand, since all sources and sinks use the same common hexagonal architecture for data storage and look up, there is no need for a flooding mechanism to track sink mobility and neither for an infrastructure setup. Therefore, the incurred delay slightly increases as the number of sink-source pair increases. The delay of LBDD and HexDD are very close to each other; however, while the number of sink-source pairs increases the difference between delay values of LBDD and HexDD gets larger since in LBDD, more sinks mean more flooding in the virtual-line resulting in increase in the traffic volume and lead to longer delivery time. In addition, greedy forwarding in LBDD may take time to find paths, more than HexDD addressing based forwarding.

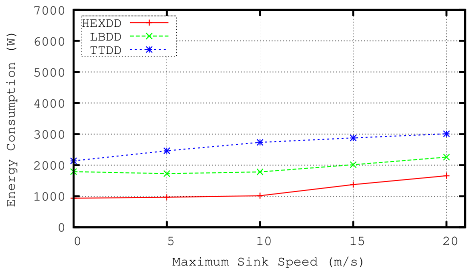
Figure 6.1(c) shows energy consumption of the whole network under each protocol. For all protocols, it is observed that the energy consumption is mainly linearly increasing with the number of sink-source pairs. TTDD presents a rather higher communication cost since there is no global virtual infrastructure in TTDD. In TTDD, every source node sends data packets to four different corners to construct its own grid structure. As the number of source node increases, a separate grid construction and maintenance on per source basis results in



(a) Data Delivery Ratio (%)



(b) Data Delivery Delay (msec)



(c) Energy Consumption (W)

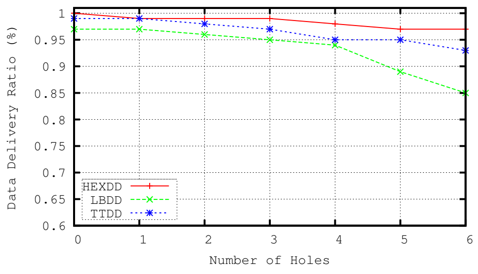
Figure 6.2: *Impact of sink speed*

higher cost, both in terms of packets and energy overhead. Also, the local query flooding mechanism in TTDD contributes significantly to overall energy consumption. Therefore, there is a big gap between plots of HexDD and TTDD in the figure, especially for higher number of sink-source pairs. The global virtual hexagonal infrastructure of HexDD results in lower energy consumptions since data packets are required to be sent only to rendezvous nodes (i.e. nodes in border cells). Although LBDD also uses a virtual infrastructure, it suffers from congestion and retransmissions during the flooding of queries through the rendezvous line.

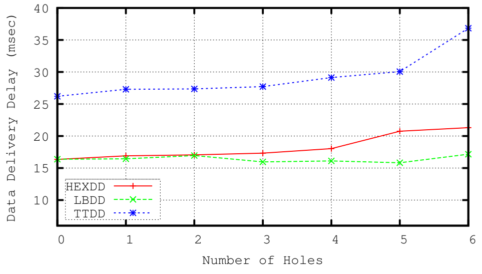
Impact of sink mobility: We have tested the performance of HexDD, LBDD and TTDD protocols under both low mobility (i.e. 4-5 km/h for walking humans) and high mobility (e.g., 50-60 km/h for UAVs) scenarios. The sinks' speeds are set to 0, 5, 10, 15, and 20 m/s (0 to 72km/h) with a pause time of 5 seconds, where the speed of 0 m/s means a static sink, obviously. The speed 20 m/s means that a sink crosses the border of a cell approximately every 7 seconds (i.e. $2r = 138.5m$ which is the longest distance in a hexagon, $138.5/20 \cong 7 seconds$) in HexDD. There are 6 sink-source pairs in this scenario.

Data delivery rate (see Figure 6.2(a)) decreases as the moving sinks' speeds increase. However, for all protocols, the success rate remains within the range of 98% to 87%. Even though there is no explicit mobility tracking scheme in HexDD, it functions well under higher mobility. HexDD performs better than the two others.

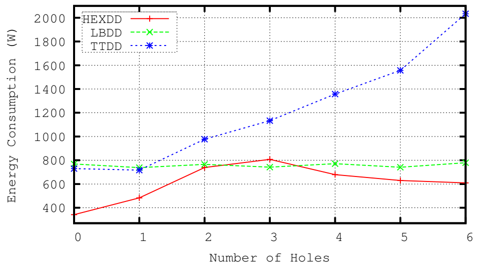
Figure 6.2(b) shows the data delivery delay vs. sinks speed. The data delivery latencies in HexDD and LBDD are lower than the delay in TTDD. Sink mobility causes the recon-



(a) Data Delivery Ratio (%)



(b) Data Delivery Delay (msec)



(c) Energy Consumption (W)

Figure 6.3: *Impact of increasing number of holes in the network*

struction of a new path between the sink and the dissemination point on the grid every time the sink changes its local cell. This results in higher delays for higher speeds of the sink due to frequent path updates. A sink can access data cached in border nodes in a short time in HexDD even while it is moving around the network. In LBDD, it has to search for data in the inline region. Therefore, LBDD has higher delay than HexDD.

Figure 6.2(c) shows that the energy consumption of TTDD is higher than HexDD and LBDD, since as the sink moves faster it tends to frequently change cells. The mobile sink renews its entire path to the dissemination point on the grid whenever it moves out of range in the local cell. Frequent renewal of the entire path to the sink increases energy consumption (see Figure 6.2(c)) and the connection loss ratio (see Figure 6.2(a)). In HexDD, the reason of slightly increasing energy consumption is also the frequent change of sinks' cell and border node which forwards the data. Due to the same reason, LBDD also does more flooding of the query for its location updates.

Impact of the number of holes in the network: Each different shaped and middle sized hole covering 5 cells in the network is randomly generated and positioned on a hextant or on the border lines. The number of sources is equal to the number of holes in the network. Each source is placed on a location where it is affected by at least one hole. Three destination sinks are chosen randomly in the network.

Figure 6.3(a) shows the impact of increasing number of holes on the data delivery ratio. We observe that for all protocols when we increase the number of holes in the network, the delivery ratio decreases. In TTDD, routing crosses over the holes for most of the data packets. TTDD does not have a specific hole recovery mechanism; however, it builds grids and its routing strategy is along the grid so if some part of the grid is missing, there are still alternative paths on the grid to forward packets towards sinks. In LBDD, on the other hand, more packets get stuck in holes when we increase the number of holes. The packets in LBDD are always directed to one single strip vertically positioned at the center of the network and for packets coming across holes, greedy forwarding sometimes fails to route across the holes. The special hole recovery mechanism in HexDD can also achive a high DDR.

Figure 6.3(b) presents the data delivery delay. TTDD has the highest delay since the alternative paths along the grids to bypass holes are in most of the cases longer than the possible shortest path between a source and a sink. Since a data packet is forwarded along the boundary of a hole in LBDD until greedy forwarding becomes possible again, it has the lowest delay. We also observe a slight decrease in the delay of LBDD between number of hole 2 and 5 on the *x* axis since when we increase the number of holes, the number of packets routed across the holes decreases. HexDD also tries to forward packets along the boundary of a hole via the shortest possible path so it also has a low delay.

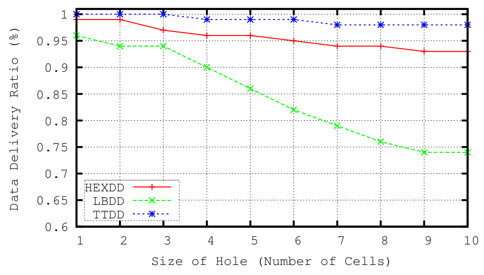
Energy consumption vs. number of holes graph is shown in Figure 6.3(c). The increase in energy consumption of TTDD is very high due to long data forwarding paths on the grid between sources and sinks to evade holes. The energy consumption of LBDD is more or less same when we increase the number of holes in the network because the protocol always tries to find a path on the border of a hole and it forwards data packet until it gets stuck at some point. Therefore, it still consumes energy on the same level although it has a lower data delivery ratio when we increase the number of holes in the network. The energy consumption of HexDD, on the other hand, increases when we have 0 to 3 holes in the network since it achieves to find paths between sources and sinks. After that point, the energy consumption decreases slightly because the data delivery ratio of HexDD also decreases slightly. Less data forwarding results in less energy consumption.

Impact of the size of holes in the network: In this set of simulations, there are one source-sink pair and one hole in the network. The size and the shape of a hole is changed for different runs. The size of the hole is represented as the number of cells that it covers.

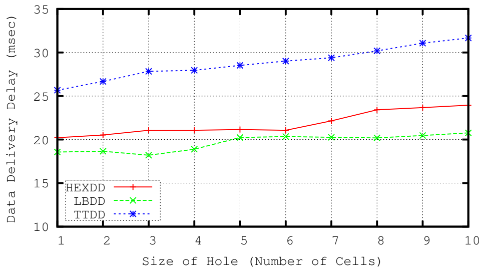
Figure 6.4(a) shows the data delivery ratio vs. increasing hole size. The grid structure of TTDD allows the data packets to find other paths to bypass a hole even when we have a large hole in the network, thus success ratio of TTDD is the highest. When we have larger holes, the delivery ratio of LBDD rapidly decreases since it is getting harder for greedy forwarding to find a path along the boundary of a large hole. More packets get stuck in the hole, e.g., 26% of the packets get lost in the hole covering 10 cells as shown in Figure 6.4(a). Simulations show that the hole recovery mechanism in HexDD works efficiently since the data delivery ratio of HexDD is high.

Figure 6.4(b) presents the data delivery delay. The delay of TTDD is again the highest, because the alternative paths along the grid maybe much more longer when passing along the boundary of a hole. Since more packets get stuck in the holes and the lost packets are not included in the delay results, the delay of LBDD is the lowest. The data delivery delay of HexDD is much more shorter than that of TTDD because HexDD tries to go along the boundary of the hole finding a shorter path.

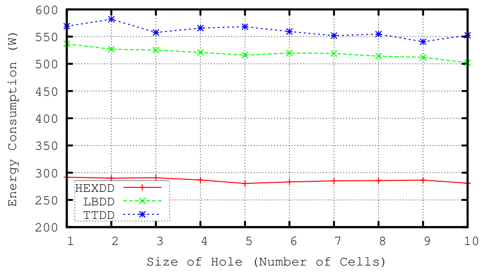
The energy consumptions of the protocols are shown in Figure 6.4(c). The energy consumption of TTDD is the highest. The energy consumption of LBDD is very close to that



(a) Data Delivery Ratio (%)



(b) Data Delivery Delay (msec)

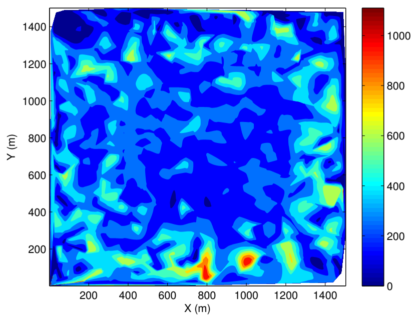


(c) Energy Consumption (W)

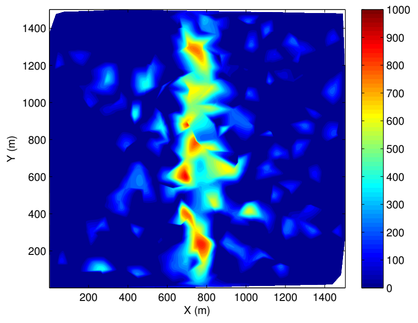
Figure 6.4: *Impact of increasing size of the hole in the network*

of TTDD although its data delivery ratio is lower than TTDD and HexDD. HexDD has the lowest energy consumption which slightly decreases due to decrease in its data delivery ratio.

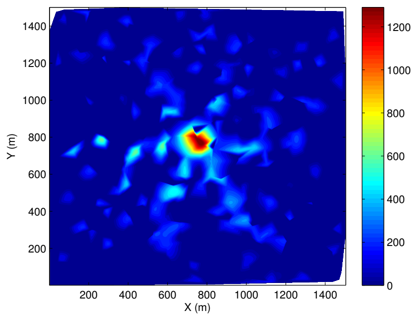
Hotspot regions: The use of a virtual infrastructure for the data dissemination can lead to the hotspot problem. Indeed, as all data reports and queries are concentrated over the rendezvous area, the hotspot problem can arise, limiting thus the network lifetime and the scalability. In this set of simulations, we analyze hotspot region in a network having 6 sources and 1 sink, which is located at different locations for three different scenarios. The simulation run lasts for 600 seconds. Figure 6.5 shows the distribution map of energy consumption for the protocols. Energy consumptions in hello packet transmissions and idle mode are not shown in the maps since it depends largely on the data generation interval and does not indicate the efficiency of the protocol. Although energy consumption is highly variable and depends on the current location of the sink and source, an important observation about our approach is that nodes in the border lines experience a higher energy consumption which shows that energy consumption is distributed among the nodes in the rendezvous region. On the other hand, the nodes close to the center of the network consume the highest energy, as expected. It is, therefore, observed that in HexDD the network lifetime is defined by a few nodes that are at the center of the network (see Figure 6.5(c)). Concerning LBDD, we notice that energy consumption is also distributed among the nodes in the rendezvous region, which is the central strip in the network. However, LBDD also has the highest energy consumption at the center of the network (see Figure 6.5(b)). In TTDD, since a separate grid structure is constructed by each individual source, the energy consumption is equally high throughout



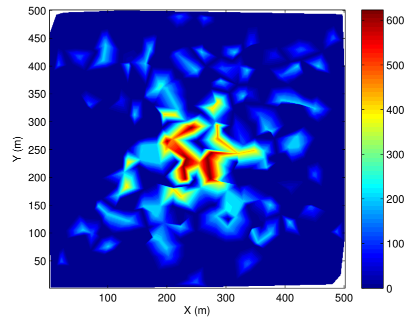
(a) Energy map of TTDD



(b) Energy map of LBDD



(c) Energy map of HexDD



(d) Energy map of HexDD with one ring

Figure 6.5: *Impact of rendezvous-based data dissemination protocols on the energy consumption distribution*

Table 6.1: *Energy Consumptions (W) of the three protocols*

	HexDD (with one cell)	**HexDD (with one ring)**	**LBDD**	**TTDD**
Overall	1611	1728	2872	9468
Maximum/node	1200	600	900	1000

the network (see Figure 6.5(a)). This increases the probability to exhaust the battery energy of the majority of nodes, leading to network partitioning and reduced network lifetime.

The overall energy consumptions of the protocols are shown in Table 6.1. The energy consumption of TTDD is higher than HexDD and LBDD, since as the sink moves it tends to reconstruct a new path to a dissemination node on the grid by local query flooding and agent updates. Also, LBDD floods the query of the sink in the inline region for its location updates. HexDD has the smallest overall energy consumption.

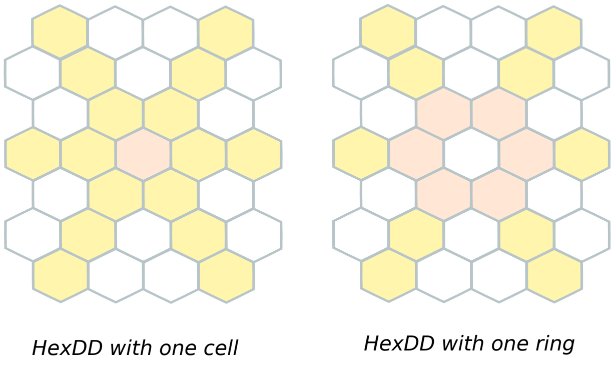
Impact of central rendezvous region size adjustment: In this section we investigate the effect of central region size adjustment on the energy distribution in the network and the network lifetime. We change the size of the central rendezvous region. Instead of having one cell at the center, we have extended the center with the first hexagonal ring as shown in Figure 6.6(a). Yellow cells are the rendezvous regions in the virtual infrastructure, and the pink cells are the central cells.

(1) Energy map of central region with one hexagonal ring: Figure 6.5(d) shows the energy distribution of HexDD with one ring approach. In this approach, the load of the central cell in HexDD with one cell approach is distributed among all the nodes in the first hexagonal ring as shown in the figure. The energy consumptions of the nodes in the first hexagonal ring cells (see Figure 6.5(d) and Table 6.1), which is around 600W, is less than the energy consumptions of the nodes in the central cell (see Figure 6.5(c)), which is around 1200W. As shown in Table 6.1, HexDD with one central cell approach consumes an overall energy of 1611W; on the other hand, the overall energy consumption of HexDD with one hexagonal ring approach, which is 1728W, is slightly higher than that of HexDD with one central cell.

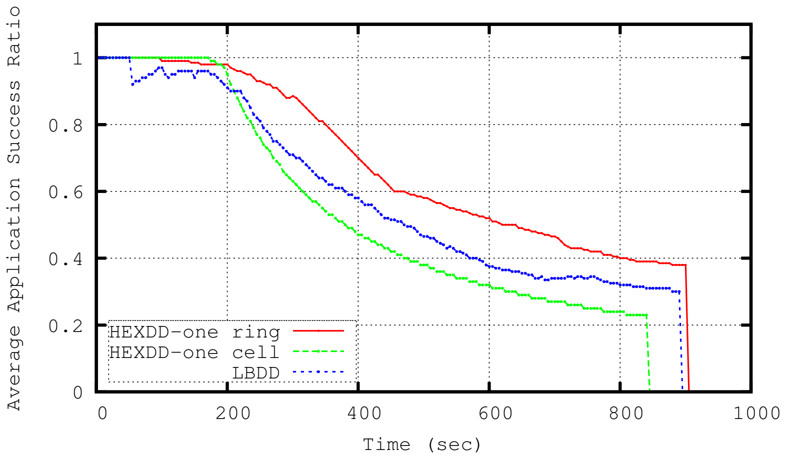
(2) Network lifetime: Several definitions of network lifetime can be found in literature. In this work, we define the network lifetime from an application point of view as *the time the application stops being operational, which in our case is the time corresponding to the last report received by a sink*. In other words, when the sink is no longer able to receive a report from the sensors, the sink is said to be disconnected from the sensors, and the network is non-functional. To analyze the network lifetime, in Figure 6.6(b) we plot the *average application success ratio*. This ratio is defined as the ratio between the total number of data reports received by the sink and the total number of reports generated by the sensors since the start of the simulation, averaged over six random network topologies.

The first phase occurs up to a duration of 200s. This phase represents the normal behavior of the network when all the sensor nodes are active. HexDD with one cell at the center presents a higher average success ratio compared to LBDD and HexDD with one ring at the center during this phase. Because LBDD and HexDD with one ring at the center have to flood the sink queries within the virtual infrastructure (i.e. line strip in LBDD and hexagonal ring in HexDD) to reach the node storing the requested data, the probability of collision is thus higher and the application is less reliable. This is why their obtained average success ratios are slightly lower than HexDD with one cell at the center. In the second phase, which occurs at an instant around 200s and the nodes start to die, LBDD and HexDD with one ring present a higher average success ratio compared to HexDD with one cell. Since they have a larger virtual infrastructure, the energy consumption of LBDD and HexDD with one ring is distributed over the entire rendezvous area, avoiding thus the hot spot problem and the existence of critical nodes such as the nodes of the center cell of HexDD with one cell at the center. Larger infrastructure introduces more redundancy between nodes, increasing the protocol robustness. This directly impacts the application success ratio which remains higher with LBDD and HexDD with one ring than with HexDD with one cell. With the above chosen metric for the network lifetime, we see that HexDD with the first-ring center lasts longer than LBDD. HexDD with larger rings can further improve the application success ratio and the network lifetime.

In the HexDD protocol, according to the network traffic, the size of the central region can be adjusted. For instance, while there is only one sink and one source in the network, only one center cell can be enough to avoid congestion at the central region. On the other hand, for larger number of sinks and sources, the central region can be extended to include the cells at the first and/or second hexagonal rings. For this adaptive mechanism, HexDD can simple check the queue sizes of the nodes in the central region. If the queue sizes of the central nodes are above a certain threshold, one more hexagonal ring joins the central region to serve as rendezvous area. When a larger central region is used, HexDD algorithm needs a small extension which enables the nodes inside the hexagonal ring to forward their packets towards the ring. This forwarding is indeed done by increasing the H until H equals to the H_c of the central ring and increasing I by k, the hextant number.



(a) Central rendezvous region size adjustment



(b) Average application success ratio

Figure 6.6: *Average application success ratio*

6.2 Vehicular Sensor Networks in Metropolitan Areas

Vehicular Sensor Networks (VSNs) are an emerging area of research that combines technologies developed in the domains of Intelligent Transport Systems (ITS) and Wireless Sensor Networks. Data dissemination is an important aspect of these networks. It enables vehicles to share relevant sensor data about accidents, traffic load, or pollution. Several protocols are proposed for vehicle to vehicle (V2V) communication, but they are prone to intermittent connectivity. In this part of the chapter we propose a roadside infrastructure to ensure stable connectivity by adding vehicle to infrastructure to the V2V communication. We adapt Hexagonal cell based Data Dissemination (HexDD) for VSNs within a metropolitan area. The virtual architecture of the proposed data dissemination protocol exploits the typical radial configuration of main roads in a city, and uses them as the basis for the communication infrastructure where data and queries are stored. The design of the communication infrastructure in accordance with the road infrastructure distributes the network data in locations that are close or easily reachable by most of the vehicles. In this section we also evaluate the performance of HexDD protocol in a vehicular sensor network scenario. The simulation results show that HexDD significantly improves the data delivery ratio in VSNs.

6.2.1 Motivating Scenario

Traditionally, towns were built in a very specific fashion. In the center would be the church or town hall and a market square, surrounded by one or more circular roads. A number of radial roads would allow visitors to travel from the city gates in the outer wall to the center. Many modern European cities reflect this old city plan in their current street layout. And still the old circular and radial roads are the main traffic arteries in the city. Figure 6.7 shows the map of the city of Enschede in the Netherlands that clearly illustrates these characteristics. If one had to choose where to build a communication infrastructure in support of vehicular networks in metropolitan areas, these roads would be the prime candidates.

In the following we adapt the HexDD protocol which provides a sensing and communication infrastructure, in support of a data dissemination protocol for vehicular sensor networks (VSNs). The main motivation is using roadside infrastructure for reliable data dissemination in VSNs. The roadside wireless sensors form a typical static Wireless Sensor Network (WSN) which provides a full and stable coverage of a city area. This WSN has advantages compared to a vehicular network whose coverage depends on the traffic situation and is usually unevenly distributed over a city. Vehicles together with roadside sensor nodes form a hybrid network that can serve many applications, such as traffic monitoring and control, environmental monitoring, and safety applications. The proposed data dissemination protocol, HexDD, can be used by these applications. We adapt the protocol for the hybrid WSN and vehicular network. In this hybrid network, thousands of vehicles are used as sensors collecting data. Data is collected in places where previously no measurements were taken, thus broadening the scale and scope of information gathering considerably. A vehicle sensor network alone has a drawback though. Dissemination of the sensor data is only possible when other cars are in communication range. If no car is in range, the data must be stored to be offloaded at a later time. The network becomes a delay tolerant network in which time between sensing the data and its dissemination can be considerable. During this period the data can become stale and not valid anymore. The use of a fixed infrastructure to offload the data

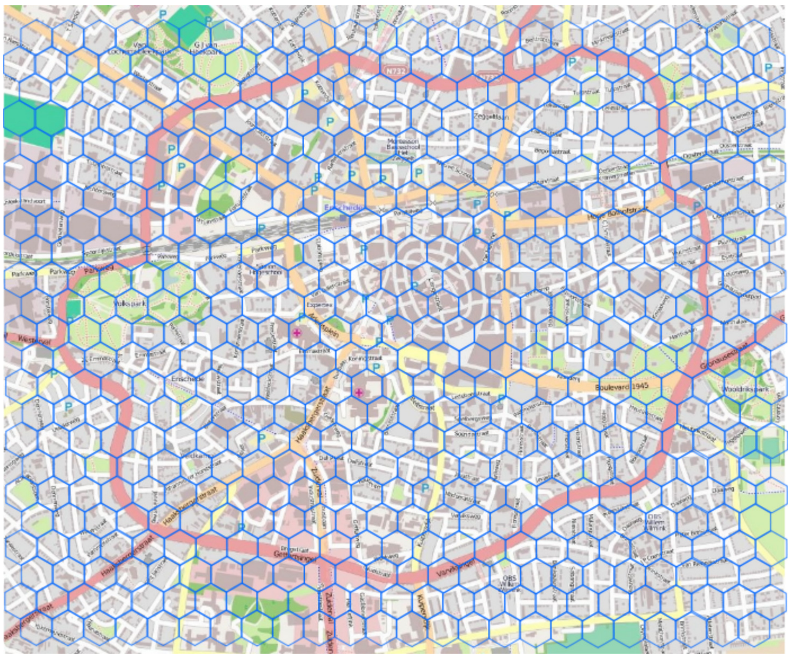


Figure 6.7: *OpenStreetMap of Enschede centre overlaid with a hexagonal tessellation of cell size around 70 meters*

to – and to get the data from as well – will improve timeliness of data dissemination.

It is tempting to demonstrate the potential of infrastructure-assisted vehicular sensor networks and HexDD with an elaborate though realistic scenario taken from one of the application areas mentioned in the following. However, for the sake of clarity we constrain ourselves in the following to a simple scenario where one vehicle provides data and another vehicle requests data.

Possible applications

Vehicular sensor networks serve as means for effectively monitoring the physical world [101]. Vehicles continuously gather, process, and disseminate relevant sensor data. Such networks allow for the emergence of several new applications. Among potential applications are:

- Traffic Monitoring and Control: Sensors deployed in both vehicles and roadside units can be used to gather information such as the speed and position of vehicles to accurately estimate the current traffic condition. Such traffic information can be combined and sent to a central authority point such as the city hall whenever requested. In addition, traffic lights equipped with sensors nodes can request live traffic information from vehicles to control the time duration of each light adaptively to the current traffic.

- Environment Monitoring: A central point can send a query for data obtained from chemical sensors, installed both in vehicles and in roadside units. Such data, combined, can provide a global estimate of the level of pollution in different regions of the city. Furthermore, sensors that are able to detect vibrations during the ride can generate estimates about the conditions of the road.

- Safety Warnings: Vehicle communication has the potential to complement internal on-board sensors (cameras or radars) to detect and warn drivers about hazardous situations when a vision beyond what sensors can provide is required. When a radio gap is present, roadside units can be used to store and later forward the corresponding data to potential interested vehicles and authorities.

6.2.2 Related Work

Vehicular Ad Hoc Networks (VANETs) are a type of Mobile Ad Hoc Networks (MANETs) used for communication among vehicles and between vehicles and roadside units (RSUs). Since the operational principles of MANETs and VANETs resemble, most of the routing algorithms that were applicable to MANETs have been considered from VANETs, and modified for their high speed mobility and the unpredictable nature of their movement. In a general context, routing protocols proposed for MANETs can be classified into two main categories: topology-based and geographical routing protocols [121]. Topology-based routing protocols exploit topological connectivity information about the network links to establish and maintain source-destination paths. In this category, protocols are mostly classified as being either proactive or reactive.

In networks utilizing a proactive routing protocol, every node maintains one or more routing tables representing the entire topology of the network. These tables are updated regularly by means of data exchange between nodes to maintain up-to-date routing information. This process can lead to a high overhead on the network. One example of a proactive protocol is the Destination-Sequenced Distance-Vector routing (DSDV) [130]. DSDV is based on the Bellman-Ford algorithm, however, with several modifications to make it suitable for a dynamic and self-starting network mechanism. In particular, it solves the routing loop problem. In the protocol, each entry in the routing table contains a sequence number generated by the destination, and the emitter needs to send out the next update with this number. Routing information is distributed between nodes by sending full dumps infrequently, and smaller incremental updates more frequently.

In contrast, reactive routing protocols only initiate a route discovery process when a route to a destination is required. This leads to a higher latency compared with proactive protocols, however, with the benefit of a lower overhead. One example of protocols in this class is the Ad Hoc On-Demand Distance Vector Routing (AODV) [131]. In AODV, a route is created on demand when a source node wants to communicate with a destination node. The route creation involves flooding a route request message and establishing, at each hop, a backward pointer (the last transmitter of the request) to the source. A reply is unicast along this path by using the backward pointers while establishing forward pointers to the destination.

In the second class of MANET protocols are the geographical routing protocols. Geographical routing relies on the geographical position of nodes to forward a packet to its destination. Because only local information is required, they do not require the establishment or maintenance of end-to-end path. In this class is the Greedy Perimeter Stateless Routing (GPSR) [89]. GPSR uses greedy geographical forwarding from the source node to the destination node. When a node cannot find a neighbor node that is closer to the destination position than itself, a recovery strategy based on planar graph traversal is applied.

Although VANETs are a special case of MANETs, the solutions proposed for MANETs do not take into account specific characteristics of vehicular environments such as intermittent connectivity and the high mobility of nodes. For these reasons, several data dissemination solutions have been proposed specifically for vehicular environments. In the remainder of this section, we review the current state-of-the-art of data dissemination protocols in VANETs by organizing recent works in two categories: *infrastructure-less* and *infrastructure-assisted*. The former comprises solutions that deal purely with vehicle-to-vehicle (V2V) communication while the latter includes solutions that make use of both vehicle-to-vehicle and vehicle-to-infrastructure (V2I) communication. Since we adapt our protocol for the hybrid of roadside wireless sensors and vehicular network, we focus on *infrastructure-assisted* protocols.

Infrastructure-assisted Data Dissemination in VANETs

The quality of services relying on vehicle-to-vehicle communication will largely depend on the available network connectivity. Therefore, especially at an initial stage of vehicular technology deployment, infrastructure will play an important role in improving the delivery ratio in sparse networks. At the time of writing of this thesis, just a few solutions have proposed the use of infrastructure to assist vehicular network protocols. In the following, we describe some of these efforts.

The use of infrastructure to improve reliability in multi-hop routing in vehicular networks was proposed in [37, 77, 152]. The work presented in [37] introduces a simple and new graph representation of the road-topology map. It takes into account the relaying capabilities of roadside units for multi-hop vehicular communications, and that can be applied to existing topology-aware routing protocols. Rather than proposing a routing protocol, authors focus on the assistance of geo-routing protocols by considering roadside units with high bandwidth, high transmission range, and all interconnected through a backbone. In [77], two novel notions are introduced to cope with link failures in vehicular networks: virtual equivalent node and differentiated reliable path. These notions are used to design the on-demand differentiated reliable routing (DRR) protocol. DRR relies on both roadside units and vehicle-to-vehicle communication to adaptively discover a sufficient number of link-disjoint paths to meet the application's specific reliability. To cope with frequent disconnections in vehicular networks, the work in [152] presents a multi-hop vehicle-to-infrastructure routing protocol, named Vertex-Based Predictive Greedy Routing (VPGR). VPGR predicts a sequence of valid vertices leveraging contextual information to forward data from a source vehicle to the infrastructure.

Also with the focus on routing, authors in [105, 129, 132] aim to improve efficiency in terms of amount of data exchanged, overhead, and energy, respectively. In [105], authors propose the multi-hop data harvesting (MDH). MDH is a data-harvesting scheme that focuses on supporting applications that require multi-hop communication, such as real-time applications. In this scheme, vehicles make use of roadside sensors to send data requests and to receive data from multiple sensors. Furthermore, a data aggregation technique is used to cope with a high amount of data when using geocasting. In [129], a novel routing approach, called RAR (Roadside-Aided Routing), is introduced. The proposed approach affiliates each vehicle to a sector, defined as the affiliation unit that is a road area bounded by neighboring roadside units (RSUs). This can reduce significantly the affiliation overhead compared to other methods that use the concept of clusters. The protocol is also based on a single-phase routing scheme. Basically, two vehicles close to each other tend to communicate directly via ad hoc networks, whereas two vehicles close to RSUs or in different sectors tend to communicate via RSUs. The roadside units are assumed to be connected with each other by wired links or any links with high bandwidth, low delay, and low bit error rate. Therefore, the routing performance is improved by limiting ad hoc routing in a small scope, and utilizing a wired backbone network.

In contrast to routing, several works have been presented with solutions for disseminating data to multiple vehicles. In [177], a data pouring and buffering paradigm for data dissemination in VANETs is proposed. Two schemes are introduced: data pouring (DP) and DP with intersection buffering (DP-IB). In DP, a data center in the infrastructure periodically broadcasts data to be disseminated and relayed by moving vehicles to pour the desired area. In DP-IB, the data poured from the source are buffered and rebroadcast at intersections. Authors in [160] consider the problem of deploying a given number of infrastructure nodes for disseminating information to vehicles in an urban area. The problem is formulated as a Maximum Coverage Problem (MCP) having as objective maximizing the number of vehicles in contact with infrastructure nodes. To provide a treatable solution, authors propose heuristic algorithms, which present different levels of complexity and knowledge.

To increase network connectivity in sparse networks, authors in [43, 110] proposed schemes with dropboxes. In [43], authors address the problem of disseminating data in sparse vehicular networks by using dead drops (dead letter boxes). Dead drops are wireless transceivers with storage capability that are not interconnected or connected to other network infrastructure. Such boxes, also known as dropboxes, can be used to both send and receive data to vehicles, in order to improve the overall network connectivity. This work presents a study of the optimum placement of dead drops in road intersections, and introduces an efficient greedy approximation algorithm called MCDD as a solution for such placement. The use of dropboxes is also discussed in [110]. Authors present a study of the impact of the following parameters when disseminating data with the help of dropboxes: end-to-end delay and packet dropping probability (PDP), by varying the number of vehicles. In the same area of research, in [109] authors tackle both the problems of limited bandwidth and minimal initial deployment. An aggregation scheme is introduced to cope with the limited bandwidth. On the other hand, by means of a genetic algorithm, the positions for placing static roadside units are identified.

A general discussion on using sensors in vehicular environments was presented in [101, 123]. In [123], authors discuss unique features and challenges that distinguish vehicular sensor networks from other types of ad hoc sensor networks. In addition, possible applications of wireless grids in addressing data aggregation and processing challenges are considered. In [101], authors survey the recent vehicular sensor network developments and identify new trends. Aspects such as how sensor information is collected, stored and harvested are evaluated considering both uses of V2V and V2I communications.

Network architectures for VSNs were subject of study in [68, 70]. The use of a hybrid ITS safety architecture is proposed in [68]. The architecture combines both vehicle-to-vehicle and vehicle-to-infrastructure sensor communication. Roadside units are connected to wireless sensor networks, thereby reducing deployment costs compared to installing dedicated roadside units. Among potential services of the hybrid communication system, the work introduces accident prevention and post-accident investigation. In addition, the main components of the system, namely, radio, networking and services, and security are described. Likewise, in [70] a similar architecture is proposed with sensor nodes deployed along the roadside to collect environmental data such as data on highway conditions (e.g. potholes, cracks on the road, ice on the road and blind spots ahead). However, the focus is on a secure data collection of such data. To achieve security, a secure symmetric key based protocol is designed and validated with real trace data through a real implementation. The work described in [143] focuses on an architecture where an ad hoc network is operated by a telecommunication provider. The goal is to combine non-valuable individual data sensed by each vehicle, in order to obtain an overview about road conditions in a certain geographical area. The aggregated information is then sent back to a roadside unit owned by the operator via a non-free frequency (WiMax or 2.5/3G). To reduce the use of high-cost links, authors present the Clustered Gathering Protocol (CGP).

With the goal of monitoring the condition of road networks, the work described in [57] presents BusNet, which is a public transport system (i.e. buses) equipped with acceleration sensors to monitor the road surface. The same application is proposed in [66]. A system referred to as the Pothole Patrol (P2) exploits the mobility of vehicles to opportunistically gather data from vibration and GPS sensors, and process the data to assess road surface conditions. By using a machine-learning approach, authors study the viability of the system to identify potholes and other road surface anomalies from accelerometer data. Related to this works is the research presented in [168]. Authors use wireless vehicular sensor networks for environmental monitoring. Experiments carried out with a sensor platform for air-quality monitoring demonstrate an improved spatial coverage when using vehicular sensors over static sensors. In [122], an open urban-scale testbed is introduced, in the effort to support novel research and application developments in wireless and vehicular sensor networks. The testbed called CitySense consists of several Linux-based embedded PCs outfitted with dual 802.11 a/b/g radios and various sensors, mounted on buildings and streetlights across the city of Cambridge.

Comparisons

Table 6.2 gives an overview of the above-mentioned works that are more similar to our proposal in this chapter. From this overview, we can outline that existing approaches propose either a routing strategy or architecture for VANET applications. There is only one combined effort of routing and infrastructure (i.e. RAR) which assumes wired links between RSUs of the backbone network. As it can be observed from the table, we can classify the routing protocols into two subclasses: (i) Geocasting, and (ii) Unicasting. In this chapter, we adopt HexDD protocol, which is a unicast routing based on location information of the vehicles and RSUs, for VSNs.

The HexDD protocol has the following advantages over the existing works for data dissemination in VSNs:

(i) It proposes the use of an inexpensive network composed of small sensor nodes to be deployed in already existing infrastructure. Such approach can decrease deployment costs compared to installing a fixed powered roadside infrastructure as proposed, for

Table 6.2: *Overview of the related works on infrastructure-assisted data dissemination in VANETs*

Name of the work	**Type**	**Goal**
DRR	Unicast routing	Multiple-path routing to meet services' reliability requirements
VPGR	Unicast routing	Context-based routing
MDH	Unicast routing & Geocasting	Data-harvesting to support applications' requirements, e.g., real-time
RAR	Unicast routing	Routing with reduced overhead by relying on a wired backbone network
DP and DP-IB	Geocasting	Dissemination to an area of interest
MCDD	Infrastructure deployment	Optimization of the placement of dead drops in road intersections
Festag et al., 2008 [68]	Architecture	Connect roadside units to wireless sensor networks to reduce deployment costs
CGP	Architecture & data aggregation	Obtain an overview about road conditions in a certain geographical area
BusNet	Opportunistic sensing	Road surface monitoring
Pothole Patrol	Opportunistic sensing	Road surface monitoring

instance, in [37, 129]. Although the use of wireless sensor networks has been proposed in [68] and [70], these works have focused on different aspects, namely, architecture and security. In contrast, HexDD focuses on routing efficiency and robustness.

(ii) Considering the advantages of using an infrastructure-assisted approach, HexDD relies on a virtual infrastructure built upon a hexagonal tessellation. Due to its optimized topology, hexagonal tessellation allows for an efficient geographical routing of event messages to any vehicle in the network.

(iii) HexDD considers end-to-end wireless communication. RSUs also communicate wirelessly via sensors attached to them.

(iv) HexDD makes the system resistant to node failures in the virtual infrastructure and supports quick routing around holes in the network.

(v) HexDD has the unique feature of leveraging the original layout of the city to build its virtual infrastructure. This allows for an improved delivery ratio and end-to-end delay.

In particular, we consider in this work the case of European cities, where circular and radial roads surrounds the city center. The layout of the virtual infrastructure is a close approximation of a city street layout, with the main diagonals being the (main) radial roads of the city and the hexagonal ring defined by the most inner ring of the city. This represents a very distinct approach when compared to other works in the current literature.

6.2.3 Infrastructure Network with Roadside Units

The vehicular sensor network that we consider is a hybrid between vehicular networks and WSNs. The network consists of static and mobile nodes. The static nodes are sensors located along the roads, attached to existing traffic signposts and other infrastructure, such as traffic lights, bus and tram stops, parking meters, railway stations, and buffer stops. Locating sensor

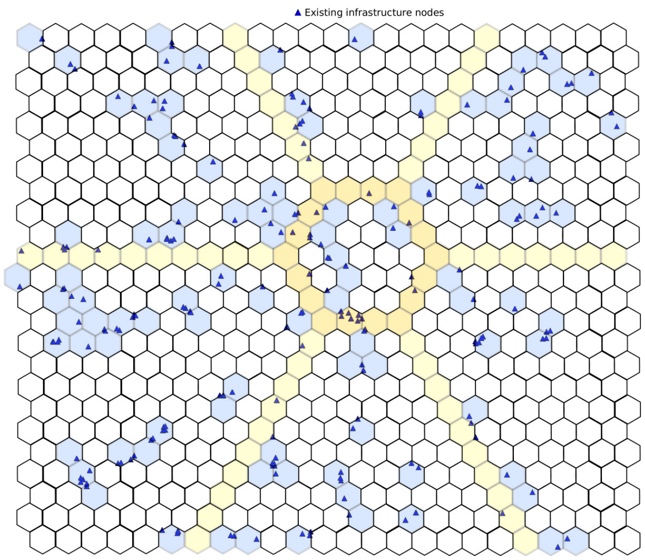


Figure 6.8: *Virtual infrastructure of the hexagonal tessellation (yellow cells) overlaid on the existing infrastructure nodes; covered cells are colored in light blue*

nodes on this kind of road infrastructure will often result in a network that is not enough dense or not evenly distributed over a city. Figure 6.8 shows the existing infrastructure nodes, the blue triangles, in the same city area shown in Figure 6.7. The distribution of these nodes is not uniform over the whole area. It is dense in some parts of the city, and sparse in some others, creating also disconnected parts in the network. Additionally, we propose to deploy small transceivers and sensors on lampposts, assuming they are regularly positioned along roads in the city, e.g. every 100 meters. Such implementation has many advantages. The lampposts are already in place and no new mechanical constructions to attach the radio nodes to are needed. Electricity is present in every lamppost and is available to power up the transceivers at minimal additional costs. Because existing utilities are used, disruptions during deployment are kept to a minimum.

These nodes embedded on roadside units (RSUs) serve as sensor and relay nodes. The network formed by them gives a complete coverage of the city area. The static nodes may be powered or able to perform energy harvesting, e.g. from sun light, thus not depending only on battery power. Vehicles moving in the city are the mobile nodes in the network. They send the information collected by their possibly many sensors to the static nodes in the network. They also ask for information from the network. Vehicles may also serve as relay nodes, passing messages from one node to another in the network, but this is more a supporting role in case holes are created in the infrastructure network.

In Figure 6.7 is shown the centre of the city of Enschede, the Netherlands overlaid with a hexagonal tessellation assuming a communication range of 250m for RSUs. Figure 6.8 shows the hexagonal tessellation for the same city area, where the light blue cells show the coverage that the exiting infrastructure nodes create for the virtual tessellation. The addition of the intelligent lampposts (iLPs) as RSUs ensures network connectivity and an acceptable sensor density, i.e. at least one sensor node per hexagonal cell. The yellow cells in Figure 6.8 show the virtual infrastructure defined by our protocol. This consists of three main diagonals of the tessellation and the n-hop ring around a centre cell (here the 3d-hop ring).

The virtual infrastructure is used by the data dissemination protocol for storing information produced by sensors in the network, e.g. detected events. Requests from vehicles are also sent to this infrastructure, making it a crucial element for the information exchange in the network. The virtual infrastructure is thus serving as a backbone for the communication. In a network where a major load of data and queries is coming from vehicles in the roads, it is reasonable to position the communication backbone in the major roads. These major roads have a strategic position for the transportation network, with most of the vehicles passing through them. The layout of our virtual infrastructure shows a strong similarity with the city street layout. This resemblance allows to use the real road infrastructure as the communication backbone in our protocol. The approach presented in this chapter is built on the premise that there is a close fit between the street layout and the virtual infrastructure. When a main road deviates from our virtual infrastructure, we use roadside units that are within the virtual infrastructure instead of main road RSUs.

6.2.4 Adaptation of HexDD for Vehicular Sensor Networks

For hexagonal tessellation construction over a city, one reference cell should be determined. A honeycomb tessellation is completely determined by one reference hexagon because, once one hexagon is known, the remaining hexagons can be easily positioned. As shown in Figure 6.9, if the center hexagon is fixed at the center of the city, the whole tessellation is fixed. In the following discussion, we assume the network has a fixed cell size, r, and network orientation. A network with a fixed cell size and network orientation is solely determined by the position of one reference cell. For node-cell association, a node needs to know the edge length of the hexagon, r, and the center of the city. To let the other far infrastructure nodes know the center of the city, a static node deployed at the center of the city can broadcast its location over the city once at the network setup phase. All the RSUs receiving this information in the city can easily associate themselves with the hexagonal cell where they are located. When a vehicle starts to move in the city or enters into a new city, it asks the network settings (i.e. cell edge size, location of the center) of this city to the nearest RSU. After getting the settings, it will be able to calculate its cell address.

In hexagonal tessellation, we classify the wireless nodes into three groups; (i) *border nodes*, (ii) *ring nodes*, and (iii) *regular nodes*, according to their position on the hexagonal tessellation. The ring cells are selected according to the position of the most inner ring of the city. If the most inner ring road of the city is covered by the hexagonal ring, g, then every node on ring g becomes a 'ring node'. In Figure 6.8, dark yellow cells are the ring cells assuming $g = 3$. The nodes associated with border cells, which are shown by light yellow cells in Figure 6.8, are called 'border nodes'. The virtual tessellation is partitioned from border cells into different parts, called 'city zones' which are the white and blue regions in Figure 6.8. All the other nodes located in city zones are called 'regular nodes'.

The virtual tessellation is partitioned from the main road-lines running through the city center into different parts (i.e. city zones) as shown in Figure 6.9. These main lines (yellow

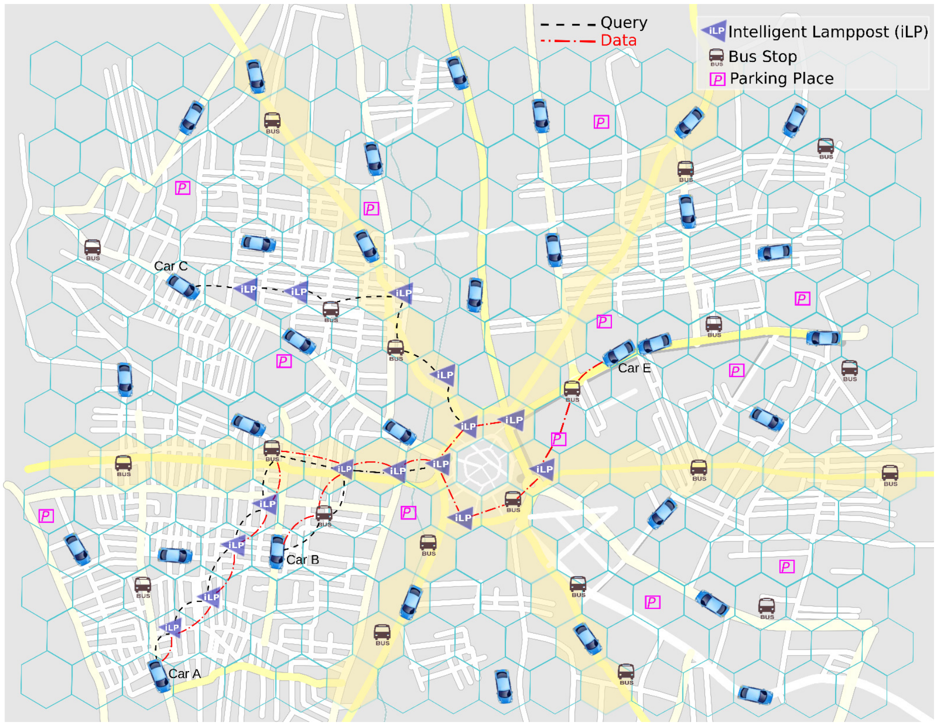


Figure 6.9: *Hexagonal tessellation overlaid on a city area (assuming g = 1) and data-query dissemination*

lines in Figure 6.9) together with a hexagonal ring (the most inner ring in a city) constitute the infrastructure for our protocol. They serve as a storage place for data and a meeting point for data and queries coming from cars moving in the city. Border nodes cache information coming from the representative zone according to the forwarding directions, which was explained in Chapter 5.3. Finally, the central ring caches the information coming from all city zones.

In the context of vehicular sensor networks, the network we envision consists of vehicles and wireless nodes located on the fixed infrastructure on the roadside. Vehicles are the mobile sources (see Car E in Figure 6.9), reporting information from collected or processed data from their possibly many sensors. They are at the same time the mobile destinations (see Car A in Figure 6.9), asking information that the driver/owner considers important. Wireless nodes embedded on roadside units (iLP, parking places (P), and bus stops in Figure 6.9) serve as sensor and relay nodes.

In the motivating VANET scenario, both the source and destination vehicles are mobile entities of the network. The impact of destination and source mobility on the dissemination scheme is very small because when destination or source vehicle moves to another cell, it only changes its connection point to the static infrastructure. When a source vehicle moves to another cell, it sends its data to the nearest RSU to become connected to the infrastructure. When destination vehicles move between cells, they need to send a new query message towards the central ring to inform the ring nodes about their new cells. If there is no direct communication between a destination/source node and a RSU, another vehicle in the next hop cell can be used as next hop until reaching a RSU. Another option is that the packet is carried by the destination/source vehicle until it could be forwarded to a node which will be a RSU, if any exits in the communication range or a vehicle. This ‘carry and forward’ concept [176] can be easily combined with our geographic forwarding protocol, HexDD.

6.2.5 Performance Evaluation in Vehicular Sensor Networks

In order to evaluate the performance of our data dissemination protocol described above, we used the open source network simulator NS-2 [15] version 2.33. We have added a new data dissemination agent (i.e. HexDD) into NS-2 over the currently implemented network stack and added our logic as a routing agent. In the following, we provide first a description of the simulation environment and scenarios characteristics and then present the evaluation methodology, the metrics for comparing the protocols. Finally, we analyze the simulation results we obtained.

Simulation Environment

In our VANET simulation we use three main components: a network component, capable of simulating the behavior of a wireless network, a vehicular traffic component, able to provide an accurate mobility model for the nodes of a VANET, and a map component, capable of creating and providing free geographic data such as street maps. The vehicular mobility and wireless network models are incorporated in different simulation tools. SUMO Simulation of Urban MObility [17] implements complex validated vehicular traffic mobility models. It is used for simulating a traffic scenario and generating an output file with vehicular mobility traces. The trace generated by SUMO is a mobility log for vehicles moving based on traffic regulations. It is possible to import different maps to SUMO to generate different test cases. Realistic urban areas (i.e. Enschede, the Netherlands) extracted from actual street maps are imported to SUMO. These maps are extracted from free maps available in OpenStreetMap [16]. After generation of mobility traces, they are fed into the network simulator, NS-2, as mobility scenario. Also, static road infrastructure points, e.g. traffic lights, transportation points, and parking meters, obtained from OpenStreetMap are used as Road Side infrastructure Units (RSUs or in other words, fixed infrastructure nodes) in NS-2. We also generated iLP nodes in NS-2. The simulation is performed by NS-2 to obtain the final results with the given inputs. Figure 6.10 shows the general view of the simulation environment.

Scenario Characteristics

In the simulations of VSNs, we consider an urban area of 3500x4000m2 that is the downtown and residential area of the city Enschede in the Netherlands. Vehicles are able to move freely on the urban graph respecting roads and intersection rules, more specifically, speed limitations and stops. Vehicles are able to communicate with each other using the IEEE 802.11 DCF MAC layer [46]. The radio transmission range has been deliberately over-evaluated and set to 250m for VANETs as we wanted to avoid biased performance evaluations due to disconnected networks. The simulation parameters are given in Table 6.3.

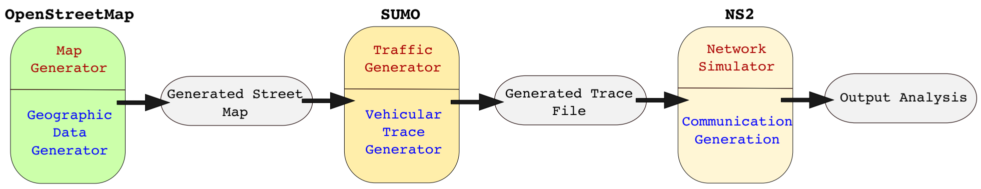


Figure 6.10: *Simulation Environment*

Evaluation Methodology

We compare the performance of the HexDD protocol with representatives from two main classes of ad hoc routing protocols: (i) AODV [131], which is a MANET reactive routing protocol, and (ii) GPSR [89], which is a MANET geographical routing protocol. AODV and GPSR protocols use both vehicles and fixed infrastructure nodes for routing, but do not yield precedence to the fixed infrastructure nodes for routing packets.

Since only a limited work has been done on infrastructure-assisted data dissemination for vehicular sensor networks inside the city environment, we have chosen two MANET protocols for comparison. Although the operations of VANET and MANET are the same, due to the difference in high speed mobility of vehicles, VANET communication requires suitable modification in the predefined routing protocols. Some efforts on improving classical MANET routing protocols to operate efficiently in VANET can be found in [25, 169]. Since we have no intention of coding these improvements in NS-2 from scratch due to time constraints, we use AODV and GPSR implementations in NS-2.33. These protocols are served as the benchmark to judge the performance of our proposed HexDD.

HexDD, AODV, and GPSR protocols are based on only local knowledge (i.e. one hop neighbors). Vehicles don't use any global knowledge such as a digital map of the region to forward their data packets [108]. In HexDD, AODV, and GPSR, we make use of periodic "hello" messages to get information from the one-hop neighbors of vehicles and RSUs.

Table 6.3: *Overall Energy Consumption (W)*

Parameters	**Values**
Simulation time	300s
Simulation area	3500x4000m^2
Transmission range	250m
Number of fixed infrastructure nodes	800 (i.e. RSUs)
Number of vehicles	300
Vehicle velocity	v_{min}=0km/h, v_{max}=100km/h
Source/Destination selection	Random
Number of source vehicles	1, 5, 10, 15, 20
Number of destination vehicles	10, 20, 30, 40, 50
MAC Protocol	IEEE 802.11 DFC
Hello Interval	1s
Data Interval	1s

Evaluation Metrics

The performance of the routing protocols has been evaluated by varying the number of destination and source vehicles. We have measured several significant metrics for data dissemination in VSNs:

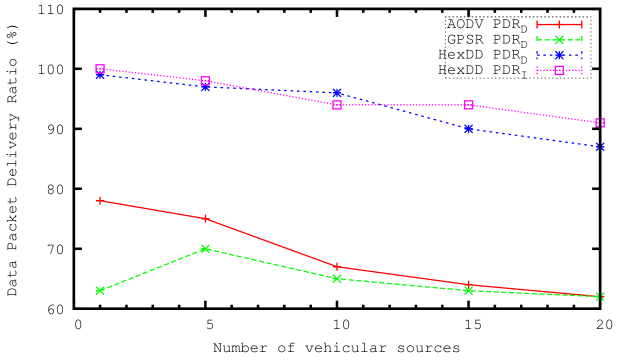
- Packet Delivery Ratio to Destinations (PDR$_D$) – It is the ratio between the number of data packets successfully delivered at destination vehicles and the number of data requests (i.e. queries) sent by the destination vehicles.
- Packet Delivery Ratio to Infrastructure (PDR$_I$) – It is the ratio between the number of data packets successfully delivered at the infrastructure and the number of data packets sent by the source vehicles. The average packet delivery ratios, PDR$_D$ and PDR$_I$, show together the ability of the routing protocol to successfully transfer data on an end-to-end basis.
- End to End Delay (E2E) – It measures the average end-to-end (i.e. source-to-sink) transmission delay by taking into account only the successfully received packets. The average delay characterizes the latency that the routing approach generated.
- Response Time (RT) – It is the average time between sending the request and getting the data for each vehicle.

Simulation Results

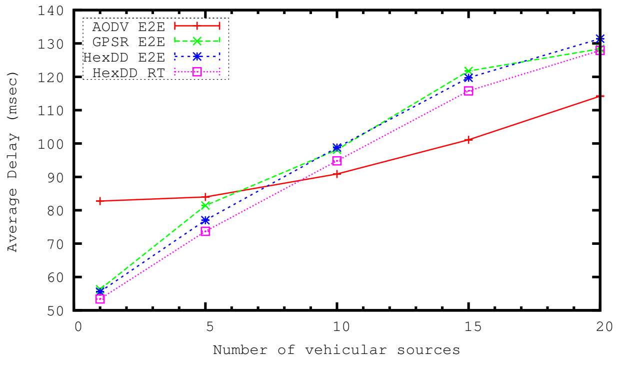
Impact of number of source vehicles: In this set of simulations, we have 30 randomly selected destination vehicles in the VANET. The graph, shown in Figure 6.11(a), demonstrates the good performance of the proposed HexDD in terms of higher PDR$_D$, compared to the other two protocols, and that is for varying number of vehicular sources in the VANET. This is an expected result of using roadside network for vehicular communication. The graph indicates that regardless of the underlying protocol, PDR$_D$ generally tends to decrease along with increase in the number of sources. Indeed, when the number of sources increases, the packet drops subsequently increase. Since only HexDD proposes a virtual infrastructure in VANET, we have calculated PDR$_I$ only for HexDD in the simualtions. The data packet delivery ratio to infrastructure is also very high in HexDD protocol. Results prove that the use of RSUs and virtual infrastructure in order to cache the data coming from different sources improves the performance of data dissemination in terms of data delivery ratio.

Figure 6.11(b) shows end to end delay for three protocols and response time for HexDD. Since we use a pull based approach in HexDD, we have also defined and measured RT, which is the time elapsed between a destination vehicle sending a query and the time it receives the data coming from the central ring. As show in the Figure 6.11(b), RT of HexDD is smaller than E2E delay of the HexDD. The E2E delay of HexDD and GPSR are very close to each other. The E2E delay of AODV is the smallest when we have 10 or more sources in the network. The AODV protocol is able to keep the average delay of the transmitted packet in an implicit control by dropping packets for which it does not have a route.

Impact of number of destination vehicles: Figure 6.12 shows the comparison of three protocols in terms of data delivery ratio and average delay for varying number of destinations when we have 20 randomly selected vehicular sources in the VANET. In Figure 6.12(a), when



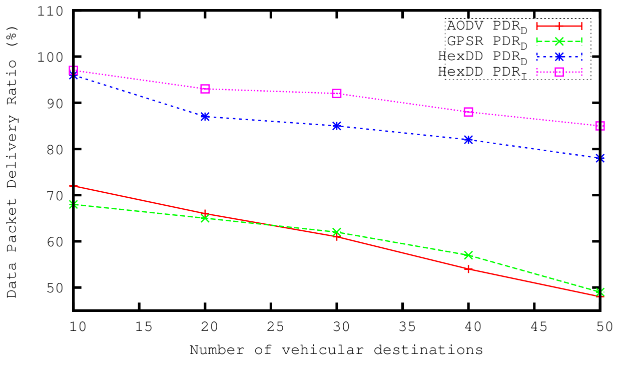
(a) Data Delivery Ratio (%)



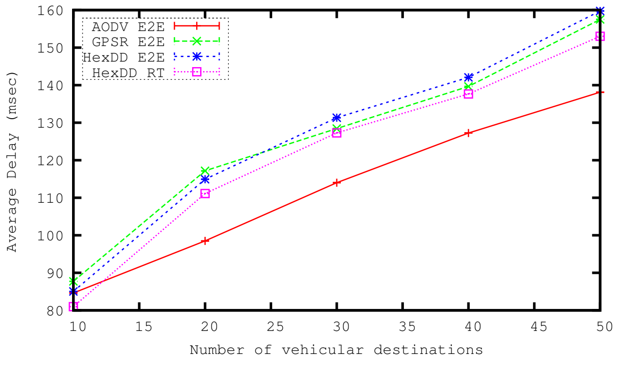
(b) Average Delay (msec)

Figure 6.11: *Performance of three protocols in terms of (a) data packet delivery and (b) average delay for different numbers of vehicular sources*

there are 10 sources in the network, PDR$_D$ and PDR$_I$ of HexDD are very close to 100%. Both PDR$_D$ and PDR$_I$ of HexDD decrease when we increase the number of destinations. However, decrease in PDR$_D$ of HexDD is bigger than decrease in PDR$_I$ of HexDD. On the other hand, AODV and GPSR show the most drastic drops in their delivery ratios, with a 20-24% decrease from the 10 destinations simulation to the 50 destinations simulation. Figure 6.12(b) plots the average data delivery delay for all protocols and also response time for HexDD. E2E delay of AODV is less sensitive to the increase of destinations than the other protocols. Since GPSR and HexDD are based on geographic routing, their E2E delays are close to each other. Both have route recovery phases when a packet reaches to a dead end. The planar graph traversal strategy of GPSR can not always guarantee to recover the route to the destination; therefore, its data delivery ratio is much smaller than HexDD. However, although the data delivery ratio of GPSR is much smaller than HexDD, its E2E delay for successfully received packets at destination vehicles is close to HexDD. This is due to the fact



(a) Data Delivery Ratio (%)



(b) Average Delay (msec)

Figure 6.12: *Performance of three protocols in terms of (a) data packet delivery and (b) average delay for different numbers of vehicular destinations*

that the route recovery strategy of GPSR also results in longer paths than recovery strategy of HexDD.

6.3 Conclusions

In the first part of this chapter, we evaluated the performance of HexDD protocol described in Chapter 5, in an emergency response scenario, which includes a mostly static deployment containing several mobile sensor nodes and mobile sinks. The simulation results demonstrate that our data dissemination strategy helps to minimize overall energy consumption and keeps the data delivery ratio high even when routing holes exist in the network. To avoid the hot region problem, which may be observed in the border lines and the central cells, one solution is to adjust the size of the border lines and shape of the central region according to the size of the network and the network traffic. In the simulations, we show the energy distribution over the network when we have different central regions (i.e. one central cell or one central ring). As recent studies has been exploiting heterogeneity in the WSNs, deployment of higher energy and communication capacity nodes can be used at the center of the network to leverage the overall system capability of HexDD. Deploying more nodes to these regions is also another solution for the hotspot problem. In [165], authors propose to deploy mobile relay nodes in hot regions. Mobile relays stay closer to the heavily loaded nodes, and take over the tasks of multiple bottleneck nodes during different network time periods. Mobile relay approach helps greatly extending the network lifetime. Our dissemination protocol can be easily incorporated with mobile relay approach.

In the second part of the chapter we adapted HexDD protocol for vehicular sensor networks and evaluated its performance in a city scenario. The network we envision consists of vehicles and roadside units. The RSUs are the lampposts equipped with small transceivers and sensors, positioned along roads at roughly equal distances, in addition to the existing communication and traffic control infrastructure. This fixed network is inexpensive, and it provides a full and stable coverage of a city area. This VSN created in this way has advantages compared to a vehicular network whose coverage depends on the traffic situation and is usually unevenly distributed over a city. This virtual infrastructure of HexDD fits with main radial roads and the inner ring of a city, which become the communication backbone for the protocol. They serve as a storage place for data and queries coming from cars moving in the city. Data and queries coming from each part of the city are sent to one of the main roads bordering it. The data is then sent towards the central ring, which has therefore knowledge about the whole city. Using main radial roads and the inner ring as rendezvous areas for data and queries, and employing roadside network for vehicular communication help to improve data delivery ratio while providing fast response in VANETs as shown in the simulations. The protocol can serve many applications of a VSN, such as traffic monitoring and control, environmental monitoring, and safety warning.

CHAPTER VII *

Conclusions and Future Work

The new generation of sensor network applications shows a clear trend of converging towards an inter-operable, heterogeneous, highly dynamic, further miniaturized form. Out of these evolutionary features, *dynamics* is an important property, which needs to be considered from the protocol design phase. The protocols designed for dynamic applications need to be well-tailored to the specific needs of the mobility of different device classes (e.g. devices of data requester, data source, data relay) of the network.

The main research focus of this thesis was to investigate how sink-to-node and node-to-sink communications could be achieved efficiently in a mobile multi-sink wireless sensor network. The overall objective of the protocols designed in this dissertation was to maximize the functionality of a *mobile multi-sink* wireless sensor network through the design of efficient, distributed and scalable algorithms. The term *efficient* can be interpreted in many ways depending on the emphasis placed by the application being considered. Although the majority of WSN-related research focuses primarily on techniques to extend the lifetime of the network, it is important to keep in mind that network lifetime is not the only issue that is of concern to the end-user. Service quality requirements, which could refer to parameters such as *packet delivery ratio* or *latency* depending on the application requirements, are definitely as well important for end-users. Hence, the research approach of this thesis incorporates the term efficiency with two aspects: quality of service and energy-efficiency. A significant proportion of this thesis is dedicated to devising strategies that would help to achieve these aspects in *mobile multi-sink* wireless sensor networks.

This thesis has taken an multi-pronged approach to address the issue of efficient data and query dissemination in mobile multi-sink wireless sensor networks. Accordingly, we provide solutions to overcome the problems brought by mobility of sensor and sink nodes. The contributions of the thesis are summarized in the following section.

7.1 Contributions Revisited

The contributions of the thesis are revisited in the following:

✓ ***Contribution 1: Benefits and challenges of using multiple sinks in static wireless sensor networks (Chapter 3):*** There are significant advantages of having *multiple sinks* in the network in terms of latency and energy consumption of information acquisition. The multi-sink partitioning of the network should be done by taking load balancing issues into account to acquire these benefits. We reviewed the state of the art load balancing methods in wireless sensor networks. We proposed a mechanism for load balancing between sinks in the network and between sensors in each partition. Although the load in the network is more balanced using the proposed balancing protocol, the bottleneck nodes near the sinks have great negative influence on the performance metrics such as data delivery ratio, throughput, and network lifetime.

✓ ***Contribution 2: Design and evaluation of a query dissemination protocol for multi-sink wireless sensor networks (Chapter 4):*** To enable sinks to efficiently route queries that are valid in particular regions of the deployment, we proposed a set of algorithms that combine coverage area reporting and geographical routing of queries injected by sinks. Each sink constructs a routing tree and collects coverage areas of itself and other nodes in the routing tree. This structure of convex hulls is used for geocasting of queries to the region of interest. Our geocasting protocol, GeoCHT, is designed for eliminating unnecessary query injections from sinks whose coverage areas do not intersect with the destination region. With this approach, we aim at decreasing energy consumption whereas meeting the requirements such as high query delivery ratio and low delivery delay. We studied the case where sinks are static and sensor nodes may be mobile. We evaluated the performance of GeoCHT protocol with extensive simulations in both static and mobile scenarios and compared its performance with another well-known geocasting protocol. Simulation results show that GeoCHT achieves an execution ratio very close to the other protocol and outperforms it in terms of network load and query delivery delay.

✓ ***Contribution 3: Handling mobility of sensors in the tree-based dissemination protocol (Chapter 4):*** To provide an up-to-date coverage area description to sinks, we have focused on handling sensor node mobility in the network. We discussed what is the best method to handle mobility in GeoCHT tree-based routing: (i) periodic global updates initiated by sinks or (ii) local updates triggered by mobile sensors. We proposed a method to perform local updates in a tree-based network. With the help of extensive simulations we observed that local updates perform very well in terms of query delivery ratio, and also more energy efficient than global updating in networks having medium mobility rate and speed, for any size of the network.

✓ ***Contribution 4: Design and evaluation of a data dissemination protocol for mobile multi-sink sensor networks (Chapter 5):*** To achieve reliable data dissemination of events as well as the efficiency in handling the mobility of multiple sinks and event sources, we proposed a virtual infrastructure and a data dissemination protocol, namely HexDD (Hexagonal cell-based Data Dissemination), exploiting this infrastructure. We analytically evaluated the communication cost and hot region traffic of the data dissemination and compared it with other approaches. Analytical comparisons show that HexDD has the lowest communication cost in both query-driven and event-driven scenarios. In addition, in most of the cases, the hot region traffic cost of HexDD is lower that other alternative approaches.

✓ ***Contribution 5: Evaluation of the data dissemination protocol for different WSN applications (Chapter 6):*** We have focused on the performance evaluation of HexDD protocol in two different classes of wireless sensor networks with *mobile* sinks: (i) Emergency Response Application – *mostly static*, containing scenarios in which most of the sensors are static and some sensors are attached to people or vehicles such as firefighters or unmanned aerial vehicles moving at low or medium velocities, (ii) Vehicular Sensor Network Application – *highly mobile*, containing scenarios in which many sensors are attached to devices that move at high velocities such as cars. We compared the performance of HexDD protocol with other application-specific protocols in terms of data delivery ratio, latency and energy efficiency. We investigated the effects of the number and speed of mobile sinks on the performance of the dissemination protocol. Moreover, we analyzed the fault tolerance performance of the protocols by varying the size and the number of holes in the network. Through extensive simulations in different application scenarios, we demonstrated that HexDD protocol provides an energy-efficient, low latency, high data delivery solution for data dissemination in mobile multi-sink WSN. On the other hand, HexDD creates hotspot regions near the center of the network. Possible solutions to the hotspot region problem were discussed.

Our hypothesis was that addressing dynamics of sink and sensor nodes in multi-sink deployments requires special attention calling for networking approaches that respond to specifics of applications. Looking back at our research objective mentioned in Chapter 1.4 and here at our contributions, it can be concluded that we have addressed both effectiveness and efficiency for routing of messages in mobile multi-sink wireless sensor networks by putting special attention to mobility handling. Therefore, we have demonstrated, with the help of several protocols presented in this thesis that our original hypothesis was valid. Having summarized the contributions, in the next section we provide a list of potential future research directions.

7.2 Future Research Directions

We believe that presence of mobility in WSNs requires special attention in networking to enhance the operations of WSNs in different application areas where mobility brings new challenges. In this thesis, we have illustrated examples of communication protocols that are designed for mobile multi-sink wireless sensor networks in order to improve the network performance. There remain some issues such as testing the presented methods on real, large scale WSN deployments. To integrate these methods with the real world, a larger scale test of the protocols presented in this thesis is desirable as part of the future work. Moreover, further improvements can be incorporated to enhance and extend the presented results. In the following we present a list of possible future directions:

- *In-network data aggregation:* Data aggregation has been widely adopted by data collection applications to reduce network traffic. In an aggregation model, a node can aggregate multiple data packets it received into one packet before relaying it. The virtual infrastructure based protocol presented in Chapter 5 can be cooperated with a dynamic in-network aggregation scheme. Since the forwarding paths along the diagonals of the sensor field are shared among all source-sink pairs in the protocol discussed in Chapter 5, it provides an opportunity for similar data to meet at some common border nodes. Data from multiple sources can be aggregated and replaced by a single data packet and forwarded towards the destined sink. Our proposed scheme can achieve further performance gain by in-network data aggregation.

- *Effects of underlying MAC protocol:* Another issue to be considered is the effect of underlying MAC protocol on the performance of the data/query dissemination. Useful enhancements for the proposed protocols can be derived by further investigating what impacts different MAC protocols have on the system. Designing a MAC protocol which can cooperate with networking layer can reduce the impact of MAC on the data/query dissemination protocols.

- *Improving efficacy of hexagonal virtual infrastructure in a real deployment:* In real life deployments, a heterogeneous network, where nodes are not the same in terms of energy, storage, and communication capacity, can improve the performance of the proposed virtual infrastructure based protocol. Higher energy and storage capacity nodes can be deployed at border lines and at the center of the network to increase the overall system capability of the protocol. Fitting of hexagonal virtual infrastructure with the real deployment area (i.e. city layout) may also improve the performance of the proposed virtual infrastructure based protocol in particular to Vehicular Sensor Network applications. A best fit can be reached via transformation of the tessellation, e.g. rotation, directional scaling, etc. Such transformation would require an adaptation of the addressing scheme, i.e. the association of a node with the virtual hexagonal cell. The routing of messages remains the same, requiring no changes.

- *Binding multiple applications on a single wireless sensor network:* Multiple applications can be invoked simultaneously on single sensor network. Triggering multiple applications on sensor networks at a post-deployment stage results into complex interactions between them [27]. Depending on the application, the characteristics of the network may dramatically change with time or space, either periodically or randomly. There are specific kinds of sinks associated with every application in a WSN. For instance, mobile and static sinks may coexist in the same network to achieve the goals of different applications. For such networks, the question ”*How can a WSN be made dynamic such that it optimizes energy-latency-load for applications with differing QoS requirements?*” should be answered to achieve the best performance.

This thesis has proposed a class of protocols and algorithms which are designed for mobile multi-sink sensor networks and are able to disseminate queries and/or data efficiently in the network with the presence of mobility. We have illustrated how mobility handling may be achieved at the network layer of the WSN protocol stack. It is hoped that the protocols designed in this dissertation can be helpful to get a few steps closer to a world where sensor networks are seamlessly embedded in the fabrics of our everyday lives.

Bibliography

Author References

[1] A. Tüysüz Erman and P. Havinga. Data dissemination of emergency messages in mobile multi-sink wireless sensor networks. In *Med-Hoc-Net'10: Proceedings of the 9th IFIP Annual Mediterranean Ad Hoc Networking Workshop*, pages 1–8, Juan Les Pins, France, June 2010. IEEE.

[2] A. Tüysüz Erman, L. van Hoesel, and P. Havinga. Enabling Mobility in Wireless Sensor Networks Cooperating with UAVs for Mission-Critical Management. In *DCOSS'08: Proceedings of the 4th IEEE/ACM International Conference on Distributed Computing in Sensor Systems (Poster Paper)*, Santorini Island, Greece, June 2008. IEEE/ACM.

[3] A. Tüysüz Erman, L. van Hoesel, P. Havinga, and J. Wu. Enabling mobility in heterogeneous wireless sensor networks cooperating with UAVs for mission-critical management. *IEEE Wireless Communications Magazine*, 15(6):38–46, 2008.

[4] A. Tüysüz Erman, T. Mutter, L. van Hoesel, and P. Havinga. A cross-layered communication protocol for load balancing in large scale multi-sink wireless sensor networks. *ISADS'09: Proceedings of the 9th International Symposium on Autonomous Decentralized Systems*, pages 1–8, March 2009. doi: 10.1109/ISADS.2009.5207344.

[5] A. Tüysüz Erman, A. Dilo, and P. Havinga. A fault-tolerant data dissemination based on Honeycomb Architecture for Mobile Multi-Sink wireless sensor networks. In *ISSNIP'10: Proceedings of 6th International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, pages 97–102, Brisbane, Australia, Dec 2010. IEEE.

[6] A. Tüysüz Erman, A. Dilo, and P. Havinga. A virtual infrastructure based on honeycomb tessellation for data dissemination in multi-sink mobile wireless sensor networks. *EURASIP Journal on Wireless Communications and Networking*, (Under Review, Submitted in April 2011), 2011.

[7] A. Tüysüz Erman, A. Dilo, L. van Hoesel, and P. Havinga. On Mobility Management in Multi-sink Sensor Networks for Geocasting of Queries. *Sensors Journal, MDPI*, (Under Review, Submitted in May 2011), 2011.

[8] A. Tüysüz Erman, R. S. Schwartz, A. Dilo, H. Scholten, and P. Havinga. *Roadside Networks for Vehicular Communications: Architectures, Applications and Test Fields*. Infrastructure Assisted Data Dissemination for Vehicular Sensor Networks in Metropolitan Areas, (Under Review, Submitted in April 2011). IGI-Global, 2011.

[9] L. van Hoesel, A. Tüysüz Erman, A. Dilo, and P. Havinga. Geo-casting of Queries Combined with Coverage Area Reporting for Wireless Sensor Networks. *Ad Hoc Networks Journal, Elsevier*, (Under Review, Submitted in July 2010, Revised and resubmitted in June 2011).

[10] L. van Hoesel, A. Tüysüz Erman, and P. Havinga. Ideas on node mobility support in schedule-based medium access. *ISSNIP'08: Proceedings of 4rd International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, pages 539–544, December 2008. doi: 10.1109/ISSNIP.2008.4762045.

[11] L. van Hoesel, A. Tüysüz Erman, and P. Havinga. Combined Coverage Area Reporting and Geographical Routing in Wireless Sensor-Actuator Networks for Cooperating with Unmanned Aerial Vehicles. In *3rd ERCIM Workshop On eMobility*, pages 43–54, Enschede, The Netherlands, May 2009. Centre for Telematics and Information Technology, University of Twente.

Web References (Last Accessed: August 2011)

[12] Camalie networks: Wireless vineyard monitoring. http://camalie.com/wirelesssensing/wirelesssensors.htm.

[13] Collaborative business items (CoBIs) funded by the European Commission, FP6 strep project. http://www.cobis-online.de.

[14] Global positioning system (gps). http://www.gps.gov/.

[15] Network simulator ns-2. http://www.isi.edu/nsnam/ns/.

[16] Openstreetmap: User-generated street maps. http://www.openstreetmap.org/.

[17] Sumo, simulation of urban mobility. http://sumo.sourceforge.net/.

[18] TinyOS. http://www.tinyos.net/.

[19] AWARE project: Platform for autonomous self-deploying and operation of wireless sensor-actuator networks cooperating with aerial objects (ist-2006-33579), 2006. http://www.aware-project.net/.

[20] GENESI project: Green sensor networks for structural monitoring, 2010. http://genesi.di.uniroma1.it/.

[21] RECONSURVE project: A reconfigurable surveillance system with smart sensors and communication, 2011. http://www.microflown-avisa.com/newsroom/news/wide-area-maritime-surveillance-reconsurve-nl.html/.

[22] D. Eberly. Intersection of linear and circular components in 2d, 2000. http://www.magic-software.com/.

[23] R.B. Muhammad. Computational geometry lecture notes, incremental convex hull algorithm. http://www.personal.kent.edu/ rmuhamma/Compgeometry/MyCG/ConvexHull/incrementCH.htm.

[24] E. W. Weisstein. Line-line intersection. from mathworld–a wolfram web resourc, 2009. http://mathworld.wolfram.com/Line-LineIntersection.html.

References

[25] Omid Abedi, Mahmood Fathy, and Jamshid Taghiloo. Enhancing AODV routing protocol using mobility parameters in VANET. In *Proceedings of the 2008 IEEE/ACS International Conference on Computer Systems and Applications*, pages 229–235. IEEE, March 2008. ISBN 978-1-4244-1967-8. doi: 10.1109/AICCSA.2008.4493539.

[26] A. A. Ahmed and N. Fisal. A real-time routing protocol with load distribution in wireless sensor networks. *Computer Communications*, 31:3190–3203, September 2008. ISSN 0140-3664. doi: 10.1016/j.comcom.2008.04.030.

[27] Ali Akbar, Ahmad Iqbal, and Ki-Hyung Kim. Binding multiple applications on wireless sensor networks. In *Advances in Grid and Pervasive Computing*, volume 3947 of *Lecture Notes in Computer Science*, pages 250–258. Springer Berlin / Heidelberg, 2006.

[28] K. Akkaya, M. Younis, and W. Youssef. Positioning of base stations in wireless sensor networks. *IEEE Communications Magazine*, 45(4):96–102, 2007.

[29] I.F. Akyildiz and M.C. Vuran. *Wireless Sensor Networks*. Ian F. Akyildiz Series in Communications and Networking. Wiley, 2010. ISBN 9780470036013. URL `http://books.google.com/books?id=7YBHYJsSmS8C`.

[30] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer networks*, 38 (4):393–422, March 2002. ISSN 13891286. doi: 10.1016/S1389-1286(01)00302-4.

[31] J.N. Al-Karaki and A.E. Kamal. Routing techniques in wireless sensor networks: a survey. *Wireless Communications, IEEE*, 11(6):6–28, 2004.

[32] M. Ali, T. Voigt, and Z.A. Uzmi. Mobility management in sensor networks. In *MSWSN: Mobility and Scalability in Wireless Sensor Networks Workshop joint with 2nd DCOSS*, pages 131–140. Citeseer, 2006.

[33] A. Baggio and K. Langendoen. Monte-carlo localization for mobile wireless sensor networks. *Elsevier's Ad Hoc Networks Journal*, 6, July 2008.

[34] M. Bahrepour, N. Meratnia, M. Poel, Z. Taghikhaki, and P.J.M. Havinga. Distributed event detection in wireless sensor networks for disaster management. In *Proceedings of International Conference on Intelligent Networking and Collaborative Systems*, INCoS '10, pages 507–512, Thessaloniki, Greece, 2010. IEEE Computer Society. ISBN 978-0-7695-4278-2.

[35] Stefano Basagni, Alessio Carosi, Emanuel Melachrinoudis, Chiara Petrioli, and Z. Maria Wang. Controlled sink mobility for prolonging wireless sensor networks lifetime. *Wireless Networks*, 14(6):831–858, February 2007. ISSN 1022-0038. doi: 10.1007/s11276-007-0017-x.

[36] A. B. Bomgni and J. F. Myoupo. An energy-efficient clique-based geocast algorithm for dense sensor networks. *Communications and Network*, 2:125–133, May 2010.

[37] D Borsetti and J Gozalvez. Infrastructure-assisted geo-routing for cooperative vehicular networks. In *Proceedings of the 2010 IEEE Vehicular Networking Conference*, pages 255–262. IEEE, December 2010. ISBN 978-1-4244-9526-9.

[38] P. Bose, P. Morin, I. Stojmenovi, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7(6):609–616, 2001.

[39] Arnab Chakrabarti, Ashutosh Sabharwal, and Behnaam Aazhang. Using predictable observer mobility for power efficient design of sensor networks. In Feng Zhao and Leonidas Guibas, editors, *Information Processing in Sensor Networks*, volume 2634 of *Lecture Notes in Computer Science*, pages 552–552. Springer Berlin / Heidelberg, 2003.

[40] S. Chatterjea. *Distributed and Self-Organizing Data Management Strategies for Wireless Sensor Networks. A Cross-Layered Approach.* PhD thesis, 2008.

[41] S. Chatterjea, T. Nieberg, N. Meratnia, and P. Havinga. A distributed and self-organizing scheduling algorithm for energy-efficient data aggregation in wireless sensor networks. *ACM Transactions on Sensor Networks*, 4:1–41, September 2008. ISSN 1550-4859. doi: http://doi.acm.org/10.1145/1387663.1387666.

[42] P. Chatterjee and N. Das. A distributed algorithm for load-balanced routing in multihop wireless sensor networks. In *Proceedings of the 9th international conference on Distributed computing and networking*, ICDCN'08, pages 332–338. Springer-Verlag, 2008. ISBN 3-540-77443-2, 978-3-540-77443-3.

[43] S.S. Chawathe. Inter-Vehicle Data Dissemination in Sparse Equipped Traffic. In *Proceedings of 2006 IEEE Intelligent Transportation Systems Conference*, pages 273–280. IEEE, 2006. ISBN 1-4244-0093-7.

[44] B. Chazelle and D. Dobkin. Decomposing a polygon into its convex parts. In *Proceedings of the 11th annual ACM symposium on Theory of computing*, STOC '79, pages 38–48, New York, NY, USA, 1979. ACM. doi: http://doi.acm.org/10.1145/800135.804396.

[45] B. Chazelle and D. Dobkin. Detection is easier than computation. In *Proceedings of the 12th annual ACM Symposium on Theory of Computing*, pages 146–153, Los Angeles, USA, 1980.

[46] Qi Chen, Daniel Jiang, Vikas Taliwal, and Luca Delgrossi. IEEE 802.11 based vehicular communication simulation design

for NS-2. In *Proceedings of the 3rd international workshop on Vehicular ad hoc networks*, pages 50–56, New York, New York, USA, 2006. ACM. doi: 10.1145/1161064.1161073.

[47] Xiao Chen and Mingwei Xu. A geographical cellular-like architecture for wireless sensor networks. In *Proceedings of the 1st International Conference on Mobile Ad-hoc and Sensor Networks*, MSN '05, pages 249–258, Wuhan, China, December 2005.

[48] Xiuzhen Cheng, Andrew Thaeler, Guoliang Xue, and Dechang Chen. TPS: A time-based positioning scheme for outdoor wireless sensor networks. In *Proceedings of the 23nd Annual Joint Conference of the IEEE Computer and Communications Societies*, INFOCOM '04, pages 2685–2696, March 2004.

[49] P. Ciciriello, L. Mottola, and G. P. Picco. Efficient routing from multiple sources to multiple sinks in wireless sensor networks. In *Proceedings of the 4th European conference on Wireless sensor networks*, EWSN'07, pages 34–50. Springer-Verlag, 2007. ISBN 978-3-540-69829-6.

[50] A. Coman, M.A. Nascimento, and J. Sander. A framework for spatio-temporal query processing over wireless sensor networks. In *Proceedings of the 1st international workshop on Data management for sensor networks (DMSN'04)*, pages 104–110, Toronto, Canada, august 2004.

[51] H. Dai and R. Han. A node-centric load balancing algorithm for wireless sensor networks. In *Proceedings of the 2003 IEEE Global Telecommunications Conference*, volume 1 of *GLOBECOM'03*, pages 548–552, dec. 2003. doi: 10.1109/GLOCOM.2003.1258297.

[52] K. Dantu, M. Rahimi, H. Shah, S. Babel, A. Dhariwal, and G.S. Sukhatme. Robomote: enabling mobility in sensor networks. In *Proceedings of 4th International Symposium on Information Processing in Sensor Networks (IPSN 2005)*, pages 404–409. IEEE, april 2005.

[53] A. Das and D. Dutta. Data acquisition in multiple-sink sensor networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 9(3):82–85, 2005.

[54] Swades De, Antonio Caruso, Tamalika Chaira, and Stefano Chessa. Bounds on hop distance in greedy routing approach in wireless ad hoc networks. *Int. Journal of Wireless and Mobile Computing*, 1(2):131–140, 2006. ISSN 1741-1084. doi: http://dx.doi.org/10.1504/IJWMC.2006.012472.

[55] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational geometry: algorithms and applications*. Springer, Berlin, Germany, 2008. ISBN ISBN 978-3-540-77973-5.

[56] S. De Vito, E. Massera, G. Burrasca, A. Di Girolamo, M. Miglietta, G. Di Francia, and D. Della Sala. Tinynose: Developing a wireless e-nose platform for distributed air quality monitoring applications. In *Proceedings of 2008 IEEE Sensors Conference*, pages 701 –704, oct. 2008. doi: 10.1109/ICSENS.2008.4716538.

[57] K. De Zoysa, C. Keppitiyagama, G. P. Seneviratne, and W.W.T. Shihan. A public transport system based sensor network for road surface condition monitoring. In *Proceedings of the 2007 workshop on Networked systems for developing regions - NSDR '07*, page 1, New York, New York, USA, 2007. ACM. ISBN 9781595937872.

[58] B. Dil, S. Dulman, and P. Havinga. Range-based localization in mobile sensor networks. In *Proceedings of the 3rd European Workshop on Wireless Sensor Networks*, volume 3868 of *Lecture Notes in Computer Science*, pages 164–179. Feb 2006.

[59] B.J. Dil and P.J.M Havinga. COM-LOC++: A distributed range-free localization algorithm in wireless networks. In *Proceedings of the 5th International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, ISSNIP '10, pages 10–11, Brisbane, Australia, 2010. ISBN 978-1-4244-7176-8.

[60] A. Dimakis, A. Sarwate, and M. Wainwright. Geographic gossip: Efficient averaging for sensor networks. *IEEE Transactions on Signal Processing*, 56(3):1205–1216, 2008.

[61] Y. Dong and D. K. Y. Yau. Adaptive sleep scheduling for energy-efficient movement-predicted wireless communication. In *Proceedings of the 13th IEEE International Conference on Network Protocols*, pages 391–400, Washington, DC, USA, 2005. ISBN 0-7695-2437-0. doi: 10.1109/ICNP.2005.6.

[62] Richard Draves, Jitendra Padhye, and Brian Zill. Comparison of routing metrics for static multi-hop wireless networks. In *Proceedings of ACM Conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM'04)*, pages 133–144, New York, New York, USA, 2004. URL `http://portal.acm.org/citation.cfm?doid=1015467.1015483`.

[63] Henri Dubois-Ferriere and Deborah Estrin. Efficient and practical query scoping in sensor networks. In *In 1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, 2004.

[64] A. Efrat, S. Har-Peled, and J.S.B. Mitchell. Approximation algorithms for two optimal location problems in sensor networks. In *Proceedings of the 2nd International Conference on Broadband Networks*, volume 1 of *BroadNets'05*, pages 714–723, oct 2005. doi: 10.1109/ICBN.2005.1589677.

[65] E. Ekici, Y. Gu, and D. Bozdag. Mobility-Based Communication in Wireless Sensor Networks. *IEEE Communications Magazine*, 44(7):56–62, September 2006. ISSN 0163-6804. doi: 10.1109/MCOM.2006.1668382.

[66] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan. The pothole patrol. In *Proceeding of the 6th international conference on Mobile systems, applications, and services - MobiSys '08*, page 29, New York, New York, USA,

2008. ACM. ISBN 9781605581392.

[67] L. Evers, M. J. J. Bijl, M. Marin-Perianu, R. S. Marin-Perianu, and P. J. M. Havinga. Wireless sensor networks and beyond: A case study on transport and logistics. Technical Report TR-CTIT-05-26, Centre for Telematics and Information Technology University of Twente, Enschede, June 2005.

[68] A. Festag, A. Hessler, R. Baldessari, L. Le, W. Zhang, and D. Westhoff. Vehicle-to-Vehicle and Road-Side sensor communication for enhanced road safety. In *Proceedings of the 9th International Conference on Intelligent Tutoring Systems (ITS08)*, 2008.

[69] C. Fischer, K. Muthukrishnan, M. Hazas, and H. Gellersen. Ultrasound-aided pedestrian dead reckoning for indoor navigation. In *Proceedings of the first ACM international workshop on Mobile entity localization and tracking in GPS-less environments, Co-located MOBICOM'08*, pages 31–36, San Francisco, USA, September 2008.

[70] H. Gao, S. Utecht, G. Patrick, G. Hsieh, F.n Xu, H. Wang, and Q. Li. High Speed Data Routing in Vehicular Sensor Networks. *Journal of Communications*, 5(3):181–188, March 2010. ISSN 1796-2021.

[71] A. Gonga, O. Landsiedel, and M. Johansson. MobiSense: Power-Efficient Micro-Mobility in Wireless Sensor Networks. In *Proceedings of the 7th IEEE International Conference on Distributed Computing in Sensor Systems, IEEE DCOSS '11*, Barcelona, Spain, 2011. URL `http://www.ee.kth.se/~oland/papers/2011-06-dcoss-gonga-mobisense.pdf`.

[72] D. H. Greene. The decomposition of polygons into convex part. *Computational Geometry, Advances in Computing Research*, 1:235–259, 1983.

[73] G. Gupta and M. Younis. Load-balanced clustering of wireless sensor networks. In *Proceedings of the 2003 IEEE International Conference on Communications*, volume 3 of *ICC'03*, pages 1848–1852, may 2003. doi: 10.1109/ICC.2003.1203919.

[74] M. Hamalainen, P. Pirinen, and Z. Shelby. Advanced wireless ict healthcare research. In *Proceedings of 16th IST Mobile and Wireless Communications Summit*, pages 1 –5, july 2007. doi: 10.1109/ISTMWC.2007.4299337.

[75] E. Ben Hamida and G. Chelius. A line-based data dissemination protocol for wireless sensor networks with mobile sink. In *Proceedings of the IEEE International Conference on Communications*, ICC '08, Beijing, China, May 2008. IEEE.

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

[77] R. He, H. Rutagemwa, and X. Shen. Differentiated Reliable Routing in Hybrid Vehicular Ad-Hoc Networks. In *Proceedings of the 2008 IEEE International Conference on Communications*, pages 2353–2358. IEEE, 2008. ISBN 978-1-4244-2075-9.

[78] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher. Range-free localization schemes for large scale sensor networks. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, MobiCom '03, pages 81–95. ACM, 2003. ISBN 1-58113-753-2. doi: http://doi.acm.org/10.1145/938985.938995.

[79] J. Heidemann, F. Silva, Y. Yu, D. Estrin, and P. Haldar. Diffusion filters as a flexible architecture for event notification in wireless sensor networks. In *Technical Report of USC/ISI, ISI-TR-556*, 2002.

[80] J. Hightower and G. Borriello. Spoton: An indoor 3d location sensing technology based on rf signal strength. In *Technical Report University of Washington*, February 2000.

[81] T.J. Hofmeijer, S.O. Dulman, P.G. Jansen, and P.J.M. Havinga. AmbientRT - real time system software support for data centric sensor networks. In *Proceedings of the 2004 Intelligent Sensors, Sensor Networks and Information Processing Conference (ISSNIP 2005)*, pages 61–66, Los Alamitos, California, USA, Dec 2004. IEEE Computer Society Press. ISBN 0-7803-8894-1.

[82] A. Howard, M. Mataric, and G. Sukhatme. Mobile Sensor Network Deployment using Potential Fields: A Distributed. In *Proceedings of the 6th International Symposium on Distributed Autonomous Robotics Systems (DARS02).*, Fukuoka, Japan, June 2002.

[83] Chong-Wei Huang and Tian-Yuan Shih. On the complexity of point-in-polygon algorithms. *Elsevier's Computers and Geosciences Journal*, 23(1):109 – 118, 1997. ISSN 0098-3004.

[84] J.H. Huang, Y.Y. Chen, Y.T. Huang, P.Y. Lin, Y.C. Chen, Y.F. Lin, S.C. Yen, Polly Huang, and L.J. Chen. Rapid Prototyping for Wildlife and Ecological Monitoring. *IEEE Systems Journal*, 4(2):198–209, June 2010. ISSN 1932-8184. doi: 10.1109/JSYST.2010.2047294.

[85] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the sixth annual international conference on Mobile computing and networking*, pages 56–67, Boston, MA USA, 2000.

[86] Aman Kansal, Arun a. Somasundara, David D. Jea, Mani B. Srivastava, and Deborah Estrin. Intelligent fluid infrastructure for embedded networks. In *Proceedings of the 2nd International Conference on Mobile systems, applications, and services - MobiSYS '04*, page 111, New York, New York, USA, 2004. ACM Press. ISBN 1581137931. doi: 10.1145/990064.990080.

[87] Holger Karl and Andreas Willig. *Protocols and Architectures for Wireless Sensor Networks.* John Wiley & Sons, Ltd, Chichester, UK, April 2005. ISBN 9780470095126. doi: 10.1002/0470095121. URL `http://doi.wiley.com/10.1002/0470095121`.

[88] Ahmed Karmouch and Sonia Hashish. Deployment-based Solution for Prolonging Network Lifetime in Sensor Networks. In *Proceedings of IFIP International Federation for Information Processing*, volume 264, pages 85–96, 2010.

[89] Brad Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, MobiCom'00, pages 243–254, Boston, MA, USA, 2000. ISBN 1-58113-197-6. doi: http://dx.doi.org/10.1145/345910.345953.

[90] M. E. Keskin, I. K. Altinel, N. Aras, and C. Ersoy. Lifetime Maximization in Wireless Sensor Networks Using a Mobile Sink with Nonzero Traveling Time. *The Computer Journal*, May 2011. ISSN 0010-4620. doi: 10.1093/comjnl/bxr048.

[91] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon. Wireless sensor networks for structural health monitoring. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, SenSys '06, pages 427–428, Boulder, Colorado, USA, 2006. ISBN 1-59593-343-3.

[92] Y. Ko and N. Vaidya. Anycasting-based protocol for geocast service in mobile ad hoc networks. *Computer Networks Journal*, 2003.

[93] Y.B. Ko and N.H. Vaidya. Location-aided routing (lar) in mobile ad hoc networks. *Wireless Networks*, 6:307–321, July 2000. ISSN 1022-0038. doi: http://dx.doi.org/10.1023/A:1019106118419.

[94] E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. In *Proceedings of 11th Canadian Conference on Computational Geometry (CCCG'99)*, pages 51–54, Vancouver, august 1999.

[95] B. Krishnamachari, D. Estrin, and S.B. Wicker. The impact of data aggregation in wireless sensor networks. In *Proceedings of 22nd International Conference on Distributed Computing Systems Workshops*, pages 575–578, December 2002. ISBN 0-7695-1588-6. doi: 10.1109/ICDCSW.2002.1030829.

[96] Ioannis Krontiris, Zinaida Benenson, Thanassis Giannetsos, Felix Freiling, and Tassos Dimitriou. Cooperative intrusion detection in wireless sensor networks. *Wireless Sensor Networks, Lecture Notes in Computer Science*, 5432:263–278, 2009.

[97] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. Worst-case optimal and average-case efficient geometric ad-hoc routing. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '03, pages 267–278, Annapolis, Maryland, USA, 2003. ISBN 1-58113-684-6. doi: http://doi.acm.org/10.1145/778415.778447.

[98] S. Lai and B. Ravindran. On distributed time-dependent shortest paths over duty-cycled wireless sensor networks. In *Proceedings of the 29th IEEE International Conference on Computer Communications*, INFOCOM'10, pages 1–9, march 2010. doi: 10.1109/INFCOM.2010.5461987.

[99] J.-J. Lee, B. Krishnamachari, and C.-C. J. Kuo. Determining Localized Tree Construction Schemes Based on Sensor Network Lifetime. *EURASIP Journal on Wireless Communications and Networking*, 2010:1–13, 2010. ISSN 1687-1472.

[100] Sang Hyuk Lee, Soobin Lee, Heecheol Song, and Hwang Soo Lee. Wireless sensor network design for tactical military applications : Remote large-scale environments. In *Proceedings of IEEE Military Communications Conference, MILCOM'09*, pages 1 –7, oct. 2009. doi: 10.1109/MILCOM.2009.5379900.

[101] U. Lee and M. Gerla. A survey of urban vehicular sensing platforms. *Computer Networks*, 54(4):527–544, March 2010. ISSN 13891286. doi: 10.1016/j.comnet.2009.07.011.

[102] X. Li, A. Nayak, and I. Stojmenovic. Chapter 6: Sink mobility in wireless sensor networks. In *Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication*. Wiley, 2010.

[103] W. Liang, J. Luo, and X. Xu. Prolonging Network Lifetime via A Controlled Mobile Sink in Wireless Sensor. In *IEEE Global Telecommunications Conference, GLOBECOM 2010*, pages 1–6, 2010. doi: 10.1109/GLOCOM.2010.5683095.

[104] W.-H. Liao, Y.-C. Tseng, K.-L. Lo, and J.-P. Sheu. Geogrid: a geocasting protocol for mobile ad hoc networks based on grid. *Journal of Internet Technologies*, 1:23–32, 2000.

[105] K.W. Lim and Y.-B. Ko. Multi-hop data harvesting in vehicular sensor networks. *IET Communications*, 4(7):768, 2010. ISSN 17518628.

[106] Ching-Ju Lin, Po-Lin Chou, and Cheng-Fu Chou. HCDD: hierarchical cluster-based data dissemination in wireless sensor networks with mobile sink. In *Proceedings of International conference on Wireless communications and mobile computing*, IWCMC '06, pages 1189–1194, Vancouver, British Columbia, Canada, 2006. ISBN 1-59593-306-9.

[107] Ren Ping Liu, Glynn Rogers, and Sihui Zhou. Honeycomb architecture for energy conservation in wireless sensor networks. In *Proceedings of the IEEE Global Telecommunications Conference*, GLOBECOM '06, pages 1–5, San Francisco, CA, USA, November 2006.

[108] C. Lochert, H. Hartenstein, J. Tian, H. Fussler, D. Hermann, and M. Mauve. A routing strategy for vehicular ad hoc networks in city environments. In *Proceedings of the 2003 IEEE Intelligent Vehicles Symposium*, volume 2000, pages 156–161. Ieee, 2003. ISBN 0780378482.

[109] C. Lochert, B. Scheuermann, C. Wewetzer, A. Luebke, and M. Mauve. Data aggregation and roadside unit placement for a VANET traffic information system. In *Proceedings of the 5th ACM international workshop on VehiculAr Inter-NETworking*, pages 58–65. ACM, 2008.

[110] M. J. Lok, B. R. Qazi, and J. M.H. Elmirghani. Data Dissemination with Drop Boxes. In *Proceedings of 2009 International*

Conference on Advanced Information Networking and Applications, pages 451–455. IEEE, 2009. ISBN 978-1-4244-4000-9.

[111] K. Lorincz, D.J. Malan, T.R.F. Fulford-Jones, A. Nawoj, A. Clavel, V. Shnayder, G. Mainland, M. Welsh, and S. Moulton. Sensor networks for emergency response: challenges and opportunities. *IEEE Pervasive Computing*, 3(4):16–23, oct.-dec. 2004. ISSN 1536-1268. doi: 10.1109/MPRV.2004.18.

[112] C.P. Low, C. Fang, J.M. Ng, and Y.H. Ang. Efficient load-balanced clustering algorithms for wireless sensor networks. *Elsevier Computer Communications Journal*, 31:750–759, March 2008. ISSN 0140-3664. doi: 10.1016/j.comcom.2007.10.020.

[113] D. Luo, D. Zuo, and X. Yang. An optimal sink selection scheme for multi-sink wireless sensor networks. In *Proceedings of the International Conference on Computer Science and Information Technology*, ICCSIT '08, pages 544 –548, september 2008. doi: 10.1109/ICCSIT.2008.161.

[114] J. Luo and J.-P. Hubaux. Joint mobility and routing for lifetime elongation in wireless sensor networks. In *Proceedings of the 24th Annual IEEE International Conference on Computer Communications (INFOCOM 2005)*, volume 3, pages 1735–1746. IEEE, 2005. ISBN 0-7803-8968-9. doi: 10.1109/INFCOM.2005.1498454.

[115] M. Ma and Y. Yang. SenCar: An Energy-Efficient Data Gathering Mechanism for Large-Scale Multihop Sensor Networks. *IEEE Transactions on Parallel and Distributed Systems*, 18(10):1476–1488, oct 2007. ISSN 1045-9219. doi: 10.1109/TPDS.2007.1070.

[116] C. Maihofer. A survey of geocast routing protocols. *IEEE Communications Surveys & Tutorials*, 6(2):32–42, 2004.

[117] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications - WSNA '02*, page 88, New York, New York, USA, 2002. ACM Press. ISBN 1581135890. doi: 10.1145/570748.570751.

[118] R. Marin-Perianu. *Wireless Sensor Networks in Motion - Clustering Algorithms for Service Discovery and Provisioning*. PhD thesis, 2008.

[119] A Milenkovic, C Otto, and E Jovanov. Wireless sensor networks for personal health monitoring: Issues and an implementation. *Computer Communications (Special issue: Wireless Sensor Networks: Performance, Reliability, Security, and Beyond)*, 29(13-14):2521–2533, August 2006. ISSN 01403664. doi: 10.1016/j.comcom.2006.02.011.

[120] Zeeshan Hameed Mir and Young-Bae Ko. A quadtree-based data dissemination protocol for wireless sensor networks with mobile sinks. In *Proceedings of 11th International Conference Personal Wireless Communications*, PWC '06, pages 447–458, Albacete, Spain, September 2006.

[121] S. Misra, I. Woungang, and S.C. Misra. *Guide to Wireless Ad Hoc Networks*. Springer-Verlag New York Inc, 2009. ISBN 1848003277.

[122] R. N. Murty, G. Mainland, I. Rose, A. R. Chowdhury, A. Gosain, J. Bers, and M. Welsh. CitySense: An Urban-Scale Wireless Sensor Network and Testbed. In *Proceeding of the 2008 IEEE Conference on Technologies for Homeland Security*, pages 583–588. IEEE, May 2008. ISBN 978-1-4244-1977-7.

[123] Maziar Nekovee. Sensor networks on the road: the promises and challenges of vehicular ad hoc networks and grids. In *Proceedings of Workshop on Ubiquitous Computing and e-Research, Edinburgh, UK*, 2005.

[124] D. Niculescu and B. Nath. Ad hoc positioning system (aps). In *IEEE Global Telecommunications Conference (GLOBECOM'01)*, volume 5, pages 2926–2931, 2001. doi: 10.1109/GLOCOM.2001.965964.

[125] G.M.P. O'hare, M.J. O'grady, D. Marsh, A.G. Ruzzelli, and R. Tynan. Autonomic Wireless Sensor Networks: Intelligent Ubiquitous Sensing (Invited paper). In *Proceeding of International Congress on Methodologies for Emerging Technologies in Automation (ANIPLA)*, volume 50, University La Sapienza, Rome, Italy, November 2006.

[126] A. Okabe, B. Boots, and K. Sugihara. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley Sons, New York, NY, USA, 1992.

[127] E.I. Oyman and C. Ersoy. Multiple sink network design problem in large scale wireless sensor networks. In *Proceeding of IEEE International Conference on Communications (ICC 2004)*, volume 6, pages 3663–3667. IEEE, 2004. ISBN 0-7803-8533-0. doi: 10.1109/ICC.2004.1313226.

[128] V. Park and M. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proceedings of the 6th IEEE INFOCOM*, Kobe, Japan, 1997.

[129] Y. Peng, Z. Abichar, and J. Chang. Roadside-Aided Routing (RAR) in Vehicular Networks. In *Proceedings of the 2006 IEEE International Conference on Communications*, volume 00, pages 3602–3607. IEEE, June 2006. ISBN 1-4244-0354-5.

[130] C.E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Proceedings of the ACM SIGCOMM Computer Communication Review*, volume 24, pages 234–244. ACM, 1994.

[131] S. Perkins, C. and Belding-Royer, E. and Das. RFC3561: Ad hoc on-demand distance vector (AODV) routing. *Internet RFCs*, 2003.

[132] Mohammad Jalil Piran. A Novel Routing Algorithm for Vehicular Sensor Networks. *Wireless Sensor Network*, 02(12): 919–923, 2010. ISSN 1945-3078.

[133] W.Y. Poe and J.B. Schmitt. Minimizing the maximum delay in wireless sensor networks by intelligent sink placement. In *Technical Report 362/07*, July 2007.

[134] W.Y. Poe and J.B. Schmitt. Self-organized sink placement in large-scale wireless sensor networks. In *Proceedings of the 17th IEEE International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems*, MASCOTS'09, pages 1 –3, sept. 2009. doi: 10.1109/MASCOT.2009.5366741.

[135] Hongliang R. and M.Q.-H. Meng. Understanding the mobility model of wireless body sensor networks. In *Proceedings of the IEEE International Conference on Information Acquisition*, pages 306 –310, aug. 2006. doi: 10.1109/ICIA.2006.306015.

[136] M. Radenkovic and B. Wietrzyk. Wireless Mobile Ad-hoc Sensor Networks for Very Large Scale Cattle Monitoring. In *Proceedings of in 6th International Workshop on Applications and Services in Wireless Networks, (ASWN'06)*, Berlin, Germany, 2006.

[137] I. Raicu, L. Schwiebert, S. Fowler, and E.K.S. Gupta. Local load balancing for globally efficient routing in wireless sensor networks. *International Journal of Distributed Sensor Networks*, 1(2):163–185, 2005. doi: doi:10.1080/15501320590966431.

[138] Ananth Rao, Sylvia Ratnasamy, Christos Papadimitriou, Scott Shenker, and Ion Stoica. Geographic routing without location information. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, MobiCom '03, pages 96–108, San Diego, CA, USA, 2003. ACM. ISBN 1-58113-753-2.

[139] Sylvia Ratnasamy, Brad Karp, and Deborah Estrin. GHT: A geographic hash table for data-centric storage. In *Proceedings of 1st ACM International Workshop of Wireless Sensor Networks and Applications*, WSNA '02, pages 78–87, Atlanta, Georgia, USA, September 2002. ACM.

[140] C. J. Ritsema, H. Kuipers, L. Kleiboer, E. Elsen van den, K. Oostindie, J. G. Wesseling, J.-W. Wolthuis, and P. Havinga. A new wireless underground network system for continuous monitoring of soil water contents. *Water Resources Research*, 45: 1–9, 2009.

[141] K. Romer and F. Mattern. The design space of wireless sensor networks. *IEEE Wireless Communications Magazine*, 11(6): 54–61, 2004.

[142] Harshavardhan Sabbineni and Krishnendu Chakrabarty. Datacollection in event-driven wireless sensor networks with mobile sinks. *International Journal of Distributed Sensor Networks*, 2010(2010), 2010. doi: http://dx.doi.org/10.1155/2010/402680.

[143] I. Salhi, M. Cherif, and S.M. Senouci. Data Collection in Vehicular Networks. In *Autonomous and Spontaneous Networks Symposium, Paris, France*, 2008.

[144] S. Schirra. How reliable are practical point-in-polygon strategies? *Lecture Notes in Computer Science, Algorithms*, 5193: 744–755, 2008.

[145] J.B. Schmitt, F.A. Zdarsky, and U.Roedig. Sensor network calculus with multiple sinks. In *Proceedings of the 11th IFIP International Conference on Personal Wireless Communications (PWC'06)*, Albacete, Spain, September 2006.

[146] K. Seada and A. Helmy. Efficient and robust geocasting protocols for sensor networks. *Elsevier Computer Communications*, 29(2):151–161, 2006.

[147] Michael Ian Shamos. *Computational geometry*. PhD Thesis, Yale University, 1978.

[148] Ahmad Sharieh, Q. Mohammad, W. Almobaideen, and Azzam Sliet. Hex-Cell: Modeling, topological properties and routing algorithm. *European journal of Scientific Research*, 22(2):457–468, 2008.

[149] C. Sharp, S. Schaffert, A. Woo, N. Sastry, C. Karlof, S. Sastry, and D. Culler. Design and implementation of a sensor network system for vehicle tracking and autonomous interception. In *Proceedings of the 2nd European Workshop on Wireless Sensor Networks*, EWSN '05, pages 93–107, Istanbul, Turkey, January – February 2005.

[150] Jeong-Hun Shin and Daeyeon Park. A virtual infrastructure for large-scale wireless sensor networks. *Computer Communications*, 30(14-15):2853–2866, 2007. ISSN 0140-3664. doi: http://dx.doi.org/10.1016/j.comcom.2007.05.020.

[151] Jeong-Hun Shin, Jaesub Kim, Keuntae Park, and Daeyeon Park. Railroad: virtual infrastructure for data dissemination in wireless sensor networks. In *Proceedings of the 2nd ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, PE-WASUN '05, pages 168–174, Montreal, Quebec, Canada, 2005. ACM. ISBN 1-59593-182-1.

[152] Raj K. Shrestha, Sangman Moh, Ilyong Chung, and Dongmin Choi. Vertex-Based Multihop Vehicle-to-Infrastructure Routing for Vehicular Ad Hoc Networks. In *Proceedings of the 43rd Hawaii International Conference on System Sciences*, pages 1–7. IEEE, 2010. ISBN 978-1-4244-5509-6.

[153] I. Stojmenovic. Geocasting with guaranteed delivery in sensor networks. *IEEE Wireless Communications Magazine*, 11(6): 29–37, 2004.

[154] I. Stojmenovic, A. P. Ruhil, and D.K. Lobiyal. Voronoi diagram and convex hull based geocasting and routing in wireless networks. In *Proceedings of IEEE Symposium on Computers and Communications (ISCC'03)*, pages 51–56, Antalya, Turkey, July 2003.

[155] Ivan Stojmenovic and Xu Lin. Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 12:1023–1032, October 2001. ISSN 1045-9219. doi:

http://dx.doi.org/10.1109/71.963415.

[156] H.-P. Tan, A.F. Gabor, W.K.G. Seah, and P.W.Q. Lee. Performance analysis of data delivery schemes for a multi-sink wireless sensor network. In *Proceedings of the 22nd International Conference on Advanced Information Networking and Applications*, AINA'08, pages 418 –425, march 2008. doi: 10.1109/AINA.2008.51.

[157] The TinyOS 2.x Working Group. Tinyos 2.0. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems (SenSys 2005)*, page 320, San Diego, California, USA, November 2005. ACM. ISBN 1-59593-054-X.

[158] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman. A taxonomy of wireless micro-sensor network models. *ACM SIGMOBILE Mobile Computing and Communications Review*, 6(2):28–36, April 2002. ISSN 15591662. doi: 10.1145/565702.565708.

[159] Godfried Toussaint. Solving geometric problems with the rotating calipers. In *Proceedings of IEEE MELECON*, Athens, Greece, 1983. URL `http://cgm.cs.mcgill.ca/~godfried/publications/calipers.ps.gz`.

[160] O. Trullols, M. Fiore, C. Casetti, C.F. Chiasserini, and J.M. Barcelo Ordinas. Planning roadside infrastructure for information dissemination in intelligent transportation systems. *Computer Communications*, 33(4):432–442, March 2010. ISSN 01403664.

[161] D. Turgut and L. Boloni. Heuristic Approaches for Transmission Scheduling in Sensor Networks with Multiple Mobile Sinks. *The Computer Journal*, 54(3):332–344, December 2009. ISSN 0010-4620. doi: 10.1093/comjnl/bxp110.

[162] L. Van Hoesel and P. Havinga. A lightweight medium access protocol (LMAC) for wireless sensor networks. In *1st Int. Workshop on Networked Sensing Systems (INSS 2004)*, 2004.

[163] L. van Hoesel and P. Havinga. A lightweight medium access protocol (LMAC) for wireless sensor networks. In *Proceedings of the 1st International Conference on Networked Sensing Systems (INSS)*, pages 205–208, Tokyo, Japan, 2004.

[164] Guiling Wang, Guohong Cao, and Thomas F. La Porta. Movement-assisted sensor deployment. *IEEE Transactions on Mobile Computing*, 5:640–652, June 2006. ISSN 1536-1233. doi: http://dx.doi.org/10.1109/TMC.2006.80.

[165] W. Wang, V. Srinivasan, and K.-C. Chua. Extending the lifetime of wireless sensor networks through mobile relays. *IEEE/ACM Transactions on Networking*, 16(5):1108–1120, October 2008. ISSN 1063-6692. doi: 10.1109/TNET.2007.906663.

[166] T. Wark, Wen Hu, P. Corke, J. Hodge, A. Keto, B. Mackey, G. Foley, P. Sikka, and M. Brunig. Springbrook: Challenges in developing a long-term, rainforest wireless sensor network. In *International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, ISSNIP'08, pages 599–604, dec. 2008. doi: 10.1109/ISSNIP.2008.4762055.

[167] M. Weiser. The Computer of the 21st Century. *Scientific American*, 265(3):94–100, 1991.

[168] K.-J. Wong, C.-C. Chua, and Q. Li. Environmental Monitoring Using Wireless Vehicular Sensor Networks. In *Proceeding of the 5th International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1–4. IEEE, September 2009. ISBN 978-1-4244-3691-0.

[169] T.Y. Wu, Y.B. Wang, and W.T. Lee. Mixing greedy and predictive approaches to improve geographic routing for VANET. *Wireless Communications and Mobile Computing*, 2010. ISSN 1530-8677. doi: 10.1002/wcm.

[170] Ya Xu, John Heidemann, and Deborah Estrin. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, MobiCom '01, pages 70–84, Rome, Italy, 2001. ACM. ISBN 1-58113-422-3.

[171] H. Yang, F. Ye, and B. Sikdar. A dynamic query-tree energy balancing protocol for sensor networks. In *Proceedings of IEEE Conference on Wireless Communications and Networking (WCNC04)*, pages 21–25, Atlanta, Georgia, USA, 2004.

[172] T. Yang, L. Barolli, G. Mino, F. Xhafa, and A. Durresi. Impact of Mobile Event Movement on the Performance of Wireless Sensor Networks. In *Proceedings of the 5th International Conference on Systems and Networks Communications*, pages 88–93. IEEE, aug 2010. ISBN 978-1-4244-7789-0. doi: 10.1109/ICSNC.2010.19.

[173] Fan Ye, Haiyun Luo, Jerry Cheng, Songwu Lu, and Lixia Zhang. A two-tier data dissemination model for large-scale wireless sensor networks. In *Proceedings of the 8th annual international conference on Mobile computing and networking*, MobiCom '02, pages 148–159, Atlanta, Georgia, USA, 2002. ACM. ISBN 1-58113-486-X.

[174] Y. Yu, R. Govindan, and D. Estrin. Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks. In *Technical Report UCLA/CSD-TR-01-0023*, University of Southern California, 2001.

[175] Y. Zhang and Q. Huang. A Learning-based Adaptive Routing Tree for Wireless Sensor Networks. *Journal of Communications*, 1(2):12–21, 2006.

[176] J. Zhao and G. Cao. VADD: Vehicle-Assisted Data Delivery in Vehicular Ad Hoc Networks. In *Proceedings of the 25th IEEE International Conference on Computer Communications*, volume 57 of *INFOCOM'06*, pages 1–12. Ieee, 2008. ISBN 1-4244-0221-2. doi: 10.1109/INFOCOM.2006.298.

[177] J. Zhao, Y. Zhang, and G. Cao. Data Pouring and Buffering on the Road: A New Data Dissemination Paradigm for Vehicular Networks. *IEEE Transactions on Vehicular Technology*, 56(6):3266–3277, November 2007. ISSN 0018-9545.

[178] J. Zheng and M.J. Lee. Chapter 4: A comprehensive performance study of IEEE 802.15.4. In *Sensor Network Operations*, pages 218–237. IEEE Press, Wiley Interscience, New Jersey, NJ, USA, 2006.

[179] H. Zimmermann. Innovations in internetworking. chapter OSI reference model – The ISO model of architecture for open systems interconnection, pages 2–9. Artech House, Inc., Norwood, MA, USA, 1988. ISBN 0-89006-337-0. URL http://portal.acm.org/citation.cfm?id=59309.59310.

[180] M. Zorzi and R.R. Rao. Geographic random forwarding (GERAF) for ad hoc and sensor networks: Multihop performance. *IEEE Transactions on Mobile Computing*, 2(4):337–348, 2003.

In pervasive systems, as they are getting smaller and smaller, computers can be found just about everywhere, but their presence is not noticed because the technologies are often embedded within items. One of the smallest and well-known embedded computers is a wireless sensor node, which is a passive sensing device capable of communicating wirelessly with other devices. Early attempts to monitor the physical environment are primarily composed of these passive sensing devices which have been succeeded in many applications by the development of Wireless Sensor Networks. A wireless sensor network is typically composed of many tiny computers, often no bigger than a coin or a credit card, that feature a low frequency processor, some flash memory for storage, a radio for short-range wireless communication, on-chip sensors and an energy source such as AA batteries. Applications of wireless sensor networks have emerged in many domains ranging from environmental monitoring to structural monitoring as well as industry manufacturing.

This thesis focuses on the efficient data extraction and dissemination in wireless sensor networks by making use of the multiple sinks and by handling mobility of sensors and sinks. We start with analyzing the characteristics of multi-sink wireless sensor networks. We propose a set of algorithms that enable multi-sink wireless sensor network to self-organize efficiently in the presence of mobility and adapt to dynamics in order to increase the functionality of the network. Our contributions include an algorithm for load balancing in multi-sink sensor network, a protocol for query dissemination towards an area of interest combined with a set of algorithms that are used to handle mobility efficiently in a tree-based routing, and a data dissemination protocol that tackles with sink mobility in a wireless sensor network.

ISBN 978-90-365-3251-8