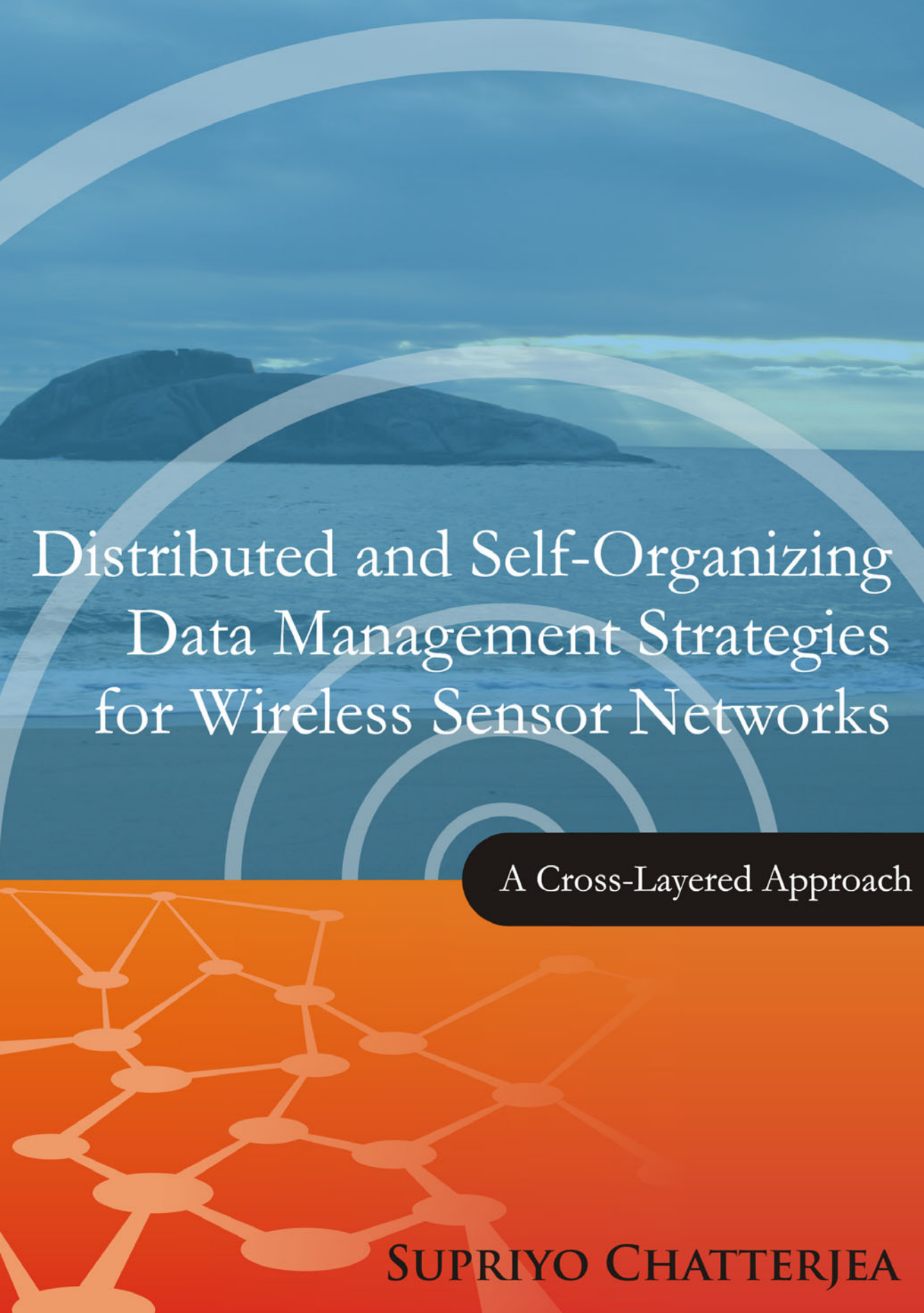
Distributed and Self-Organizing
Data Management Strategies
for Wireless Sensor Networks
A Cross-Layered Approach
Supriyo Chatterjea

Distributed and Self-Organizing Data Management Strategies for Wireless Sensor Networks

A Cross-Layered Approach

Supriyo Chatterjea

Composition of the Graduation Committee:

Prof. Dr. Ir.	G.J.M.	Smit	(UT, CAES)
Dr.	P.J.M.	Havinga	(UT, PS)
Prof. Dr.	H.	Brinksma	(UT, PS)
Prof. Dr. Ir.	Th.	Krol	(UT, CAES)
Prof. Dr.	J.J.	Lukkien	(TU Eindhoven)
Prof. Dr.	I.W.	Marshall	(Lancaster University, United Kingdom)
Dr.	M.	Palaniswami	(University of Melbourne, Australia)

University of Twente
The Netherlands

EWI/PS
P.O. Box 217, 7500 AE Enschede
The Netherlands.

NWO
Netherlands Organisation for Scientific Research

CTIT
Centre for Telematics and Information Technology

This research has been funded by NWO (The Netherlands Organisation for Scientific Research) and has been carried out within the context of the Center for Telematics and Information Technology (CTIT).

This thesis was edited with WinEdt and typeset with LaTeX2e.
Keywords: Wireless Sensor Networks, Distributed, Self-Organizing, Cross-layered.

Cover Design: Supriyo Chatterjea; AIMS Beach, Townsville, Australia

Printed by Wöhrmann Print Service.

ISBN 978090-365-2721-7
CTIT PhD Thesis Series Number 08-127

DISTRIBUTED AND SELF-ORGANIZING DATA MANAGEMENT STRATEGIES FOR WIRELESS SENSORS NETWORKS

A CROSS-LAYERED APPROACH

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof. dr. W.H.M. Zijm,
volgens besluit van het College voor Promoties
in het openbaar te verdedigen
op vrijdag 26 september 2008 om 13.15 uur

door

Supriyo Chatterjea

geboren op 15 maart 1976

te Kolkata, India.

Dit proefschrift is goedgekeurd door

Prof. Dr. Ir.	G.J.M	Smit	(promotor)
Dr.	P.J.M	Havinga	(assistent-promotor)

Abstract

Over the past few decades the computing industry has gone past several milestones, each of which has had a paradigm shift in the way we live our lives. Computers were initially only confined to large corporations. With the advent of personal computers, people started using them daily at work and also at home. Today it is not uncommon for a person to move around with several devices which have substantial amounts of computation power, e.g. a notebook, mobile phone, PDA, digital camera, navigation system, etc. While one may be inclined to feel that we are already surrounded by a huge number of embedded devices, it appears that the computing industry, thanks to the further miniaturization of electronics, might be on the verge of another paradigm shift - one that would make computers omnipresent. The past decade has seen the emergence of a new breed of tiny computers known as *wireless sensor nodes*. These nodes, which may be battery powered, are equipped with sensors, a radio transceiver, a CPU and some memory. They are usually networked together to form *wireless sensor networks*. It is envisioned that sensor networks made up of hundreds, thousands or probably even millions of nodes will eventually weave into the very fabric of our lives and be present in one form or another in even the most mundane of devices, like in a coffee mug for instance.

The enormous scale of these networks makes it impossible for them to be managed manually by humans. In other words, the system needs to operate autonomously and recover automatically from faults that may occur. As sensor nodes are typically highly energy constrained devices, network lifetime is also of paramount importance. Unlike conventional computer networks, e.g. an office LAN, which can be used for a multitude of applications, wireless sensor networks are generally known to be application-specific. This unique characteristic helps save energy as it allows protocols designed for sensor networks to be optimized for a particular application.

This thesis focuses on different techniques that may be used to extract data

from a wireless sensor network in an energy-efficient manner. We present a range of distributed, self-organizing and energy-efficient data management algorithms that influence different components of the sensor node architecture: MAC, routing, data aggregation and sensor sampling.

The algorithms we present are used for two different classes of applications: (i) applications that only require a subset of all the data in the network to be extracted using *range-queries* and (ii) applications that require all the data to be extracted from all the sensors in the network at periodic intervals using *long-running queries*.

For the first class of applications, we present a framework that ensures that one-shot range queries are routed only to the relevant regions of the network instead of carrying out flooding. The same framework is also used to assign an appropriate amount of bandwidth to regions that are expected to generate more data with respect to the incoming query. This allows precious energy-resources to be spent only where gains are expected. For the second class of applications, we have designed two algorithms that help extract raw data in an energy-efficient manner. The first algorithm, that takes advantage of spatial correlations that may exist between the readings of neighbouring sensor nodes, is a scheduling algorithm which decides when a particular node should aggregate data. The second algorithm helps save energy by sampling the sensors in an energy-efficient manner by taking advantage of temporal correlations that may exist between successive sensor readings. In every instance, we have illustrated how the various algorithms can benefit by using cross-layer information.

Samenvatting

In de afgelopen decennia heeft de computer industrie enkele mijlpalen bereikt, die elk een grote invloed hebben gehad op onze manier van leven. Eerst waren computers alleen beschikbaar voor grote bedrijven. Met de introductie van de PC, begonnen mensen dagelijks, zowel op het werk als thuis, de computer te gebruiken. Vandaag de dag is het niet ongewoon dat men meerdere apparaten op zak heeft, die elk veel rekenkracht hebben. Voorbeelden zijn een notebook, een mobiele telefoon, een zakagenda, digitale camera, een navigatie systeem, etc. Hoewel het lijkt dat we al door veel van dit soort apparaten worden omringd, is de computer industrie -dankzij verdere miniaturisatie van elektronica- bezig met de ontwikkeling van een nieuw fenomeen: de alom aanwezige computer. In de afgelopen jaren is een nieuw soort computer ontwikkeld, bekend als draadloze sensoren. Deze computers, die door een batterij gevoed kunnen worden, zijn uitgerust met sensoren, een radio zender en ontvanger, een processor en wat geheugen. Deze apparaatjes vormen normaal gesproken een draadloos sensor netwerk met elkaar. Zulke netwerken kunnen uit honderden, duizenden of misschien wel miljoenen draadloze sensoren bestaan en zullen zich mengen met onze dagelijkse activiteiten. Zo zullen de kleine computertjes gentegreerd worden in alledaagse dingen, zoals een koffiemok.

De gigantische schaal van deze netwerken maakt het onmogelijk om ze handmatig te bedienen. Met andere woorden, het systeem moet autonoom kunnen functioneren en als er fouten optreden, moet het zichzelf kunnen herstellen. Hoewel de draadloze sensoren een zeer beperkte energievoorraad hebben, is een lange levensduur van het netwerk uiterst belangrijk. In tegenstelling tot conventionele computer netwerken (bijvoorbeeld een LAN in een kantooromgeving), die voor vele verschillende toepassingen kunnen worden gebruikt, zijn draadloze sensor netwerken applicatiespecifiek. Deze unieke eigenschap maakt het mogelijk om energie te besparen, omdat se sensor netwerk protocollen kunnen worden geoptimaliseerd voor de specifieke toepassing.

Deze dissertatie richt zich op verschillende technieken, die gebruikt kunnen worden om op energie-efficiënte wijze data uit draadloze sensor netwerk te verkrijgen. We presenteren verschillende gedistribueerde, zelforganiserende en energie-efficinte data management algoritmes, die ingrijpen in verschillende onderdelen van de draadloze sensor architectuur: MAC, routing, informatie combinatie/representatie en sensor bemonstering.

De gepresenteerde algoritmes kunnen voor twee toepassingsklasse gebruikt worden: (i) toepassingen die slechts een deel van alle informatie uit het netwerk gebruiken en dat door middel van "range-queries" opvragen en (ii) toepassingen die periodiek alle informatie uit het netwerk nodig hebben en die dat met langgeldende queries opvragen.

Voor de eerste categorie toepassingen presenteren we een architectuur waarin eenmalige "range-queries" alleen verstuurd worden naar relevante gebieden in het netwerk in plaats van naar alle sensoren in het netwerk. Dezelfde methode wordt gebruikt om de relevante gebieden in het netwerk meer bandbreedte toe te kennen, mocht dat voor de query nodig zijn . Schaarse energie wordt op deze manier alleen maar besteed waar het het meeste oplevert.

Voor de tweede toepassingsklasse zijn twee algoritmes ontwikkeld, die het energie-efficint extraheren van ruwe data vergemakkelijken. Het eerste algoritme haalt zijn voordeel uit (ruimtelijke) correlaties die kunnen bestaan uit metingen van naburige draadloze sensoren. Het algoritme bepaalt wanneer er (gecorreleerde) informatie door een draadloze sensor moet worden samengevoegd. Ook kunnen er correlaties bestaan tussen opeenvolgende bemonsteringen van een sensor. Het tweede algoritme zorgt dat er energie bespaard wordt op het bemonsteren van sensoren door correlaties in het tijddomein uit te buiten. Bij elk van de algoritmes tonen we aan dat het gebruik van cross-layer informatie voordelig is.

Acknowledgements

What an adventure it has been! Experiencing endless simulation crashes, camping overnight in office, dodging category 5 cyclones and blood-sucking sand flies, rushing for paper deadlines into the wee hours of the morning, spending hours under the shower pondering about protocol design, penning down dozens of project proposals, "turbo-boost"-ing over dead kangaroos Down Under (in a 800cc Daihatsu Charade) - and surviving (!), nearly being labeled as an international terrorist who uses sensor nodes for eaves dropping and "RSI-ing" myself nearly from head to toe during the thesis write-up phase. This is probably the closest a computer scientist can get to living the life of Indiana Jones! And now, finally, the time has come for me to write the acknowledgements, which probably is the most important part of this book. After all, this adventure would not have been even half as enjoyable, memorable and exciting, had it not been for the wonderful people I have been privileged to meet during the course of my PhD.

Over the past few years, I have had the opportunity to work under two promoters in succession, Thijs Krol and Gerard Smit. Although I did not have much interaction with them on a daily basis, they both played crucial roles.

In the midst of my second year, when I was wrestling desperately with the IND on a daily basis to bring my wife, Anindita, over to join me here in the Netherlands, Thijs' letter to the IND worked like a charm and definitely helped speed up the application process. Had it not been for his help, I am certain I would have been forced to spend a few more months of my life as a bachelor with huge telephone bills.

Gerard, though not from the field of sensor networks himself, had a long list of very useful comments that really improved the quality of this thesis. I really appreciate the fact that he read my thesis in such great detail.

I consider myself extremely fortunate to have had Paul Havinga as my daily

supervisor. Right from the beginning, Paul has always been extremely supportive and approachable. I can't remember any instance when he didn't have time for me. No matter how busy he was, not once was I shooed away. One of the things that I really appreciated was that Paul treated his PhD students, not simply as students but rather, as equal members of the group. It is this lack of hierarchy that allows one to try out things that are beyond the usual domain of a PhD student and as a result I was introduced into the world of project proposal writing. While it did eat up into my time dedicated to research, I learnt an important skill that I'm sure will help me later in life. I'm really grateful to Paul for providing me with such opportunities. He is also very flexible - if he finds that you're really passionate about something, he always allows you to do it and in my opinion, that's a great way to get the best out of people. However, there are times when he makes sure he has his way: I still don't get why he has this unwritten rule about not being allowed to submit papers to conferences held in Hawaii! Somehow he never bought the argument that visiting exotic places like Hawaii, will result in the generation of more creative research ideas which translates into better publications and thus a higher citation record! I guess I need to work on coming up with something more convincing...

I would also like to thank the Netherlands Organisation for Scientific Research (NWO) for funding my research here at Twente through the Consensus project. Had it not been for them, I would not have had the opportunity to carry out research in the Netherlands. The EU-funded Sensei project also funded part of my research.

Many thanks also to the members of my graduation committee. Their insightful comments have definitely improved the quality of this thesis.

In the third year of my PhD I had the opportunity to visit the Australian Institute of Marine Science (AIMS) in Townsville, Australia for a period of three months. I cannot imagine what my stay in Townsville would have been like without the help of Stuart Kininmonth from AIMS. Stuart's warmth and hospitality, right from the very minute we first met, is something I will never forget. He never hesitated to offer his help with network deployments even on weekends. It's also due to his efforts that we are still continuing to collaborate on our deployments on the Great Barrier Reef. Of course, in return I've had to put up with his comments about me being a "Singaporean wimp", simply because when it came to deploying sensor nodes deep inside bushes, riddled with cobwebs, I would leave it to him. But you can hardly blame me when you consider the fact that of the 9 most poisonous arachnids, Australia has all of them. Besides, if you were an Australia funnel-web spider and you saw some

goon placing a gigantic PCB with blinking lights (i.e. sensor node) right in the midst of your intricate spider web, would you rather sink your fangs into: a 1.76m homo sapien or a 2m Scottish-Australian hulk of a guy who cycles 100km a day to and from work? It's an obvious choice!

I would also like to thank Leon from Ambient Systems. I spammed him with dozens of emails asking him for help when I was in Australia, but, he always made a point to reply to every one in great detail.

I have shared my office room with different sets of people over the years. I started off with the EYES gang: Stefan, Tim, Lodewijk and Jian - and later, Tjerk. I'll always remember our infamous daily dart games which helped prove one famous theory wrong: Practice makes perfect! While our skills did improve over the first year, mysteriously the scores began to plummet after that. We tried to blame it on the equipment - but a new set of darts didn't help either! Moral of the story? Pratice makes one perfect but too much of any good thing, even if it is practice, leads to problems!

Stefan's ever helpful nature and Tim's eye for perfection (like any good mathematician) were commendable. Lodewijk was always keen on sharing his knowledge on anything Dutch, be it cycling routes or Dutch language. In fact he along with Tjerk took the trouble to provide daily Dutch lessons once the Smart Surroundings folks joined us. All these pleasant memories will be firmly etched in my mind.

I was also fortunate to have a great bunch of friends outside my own office room - Nikolay, Ricardo, Laura and Law are all fantastic human beings who really spiced things up along our corridor - who said computer scientists are boring nerds??!! Nikolay - whom I assumed to be an ex-Olympian weightlifter cum computer scientist during our first encounter - was the one who introduced me to Enschede. Of course I had no clue then that Bulgarian weightlifters get absolutely petrified when Dutch ghosts come knocking on your wall. I had several parties at home where Ricardo and Laura dazzled us with their dancing skills. They are an extremely sweet couple - but they're definitely not even half as sweet as cute little Paulina. I'll always remember Law - one of the three Malaysians in our department (which meant a problem since I was outnumbered 1:3) - for his dry humour.

The start of the Smart Surroundings project suddenly brought a bunch of smart (and extremely nice) people to the room next to ours. Mihai, Raluca and Ozlem have been great friends who are also wonderful to work with. I'll always remember how the four of us ran off to Venice (without telling Paul of course) on the way to Los Angeles. It really helps to go on conferences without the

supervisor around, isn't it? The most wonderful thing about our group is its diversity. And this gave us the opportunity to excite our palette by trying out traditional food from different countries? Of course my dear Romanian friends (Mihai, Raluca, Stefan and Ileana) may tend to disagree as even the most mildly cooked Indian food seems to cause their tongues third degree burns.

During the last part of my PhD I had two new office mates: Yang and Aysegul. Many thanks to Yang for all his help with the implementation work. Very few people I know can put in the kind of hard work that he's able to do. He worked tirelessly even throughout the weekends and I appreciate his help tremendously. It's been great having you guys as room mates.

It's been a pleasure working with Nirvana on the numerous project proposals. One thing I noticed working with her, is that it's nearly impossible to get stressed up when she is around, no matter how tight the deadline maybe. Her infectious laugh always helps keep your blood pressure at the optimal level. Nirvana really tries to help out, even if it inconveniences her. The help she rendered when Anindita was recovering from her unfortunate accident or when I was just about to leave for Australia are things which I'll never forget. Maria too is always ready to help - be it about the taxation system of the Netherlands or child care stuff, Maria's a one stop information centre for all the important things you need to know. Of course our conversations about Nicole's long line of curious questions are a constant source of enjoyment and amazement.

Our secretaries, Marlous, Nicole and Thelma have made a world of a difference to my stay so far in our group. I don't think it's possible to find a bunch of secretaries who are more helpful than them. But it is not just their helpfulness that I'll remember. All three of them are extremely warm individuals... Every once in a while, when I used to get bored sitting at the computer, chatting with the three of them was always delightful. But besides entertainment they're a storehouse of knowledge. From where to get the best deals on baby products to names of "cesar" therapists, or simply exchanging tips on holiday experiences - chatting with Marlous, Nicole or Thelma is always a refreshing break from the monotony of sitting at the desk.

I dread to think what my life would have been like over the past few years had it not been for the presence of my Indian friends at the UT. They have been like an extended family to both Anindita and me. Such warmth is usually only received from the closest of your own relatives, but coming to the UT, I've realised that it is possible to have friends who genuinely care about you. Vasughi, Vijay and Sheela were among the first Indians I met on campus. It was Vijay's constant company towards the beginning and his bubbly personality

that made me forget my homesickness. Both he and Vasu made sure I was well-introduced to the Indian community on campus and it was solely because of them, that I never felt lonely even though I was thousands of miles away from home. I'll never forget our regular weekend movie nights at Vasu's place together with Vijay, Sheela, Komal, Manish, Vishy and Deepa. I thoroughly enjoyed the daily cricket matches in the evenings with Ravi and Madhavi, Salim, Pramod and Shankar even though my rotten cricketing skills must have irritated quite a few! A few months after residing at Calslaan Blue, I came to know my neighbour - Jay Kolhatkar. He's an extraordinary person intelligent, warm and well informed on a variety of issues. I remember our conversations during our frequent biking trips during weekends which spanned topics ranging from politics to history to statistics and of course on how to get hitched to the right person. One thing's for sure. It wasn't all just talk - we both managed to find our better halves. So it sure was time well spent!

There were several difficult times over the past few years when I initially wished I was back in Singapore in the comfort of home - for example, when I had my wisdom tooth surgery and when Anindita was involved in the horrible accident. In both cases, my Indian friends were always there without fail to give us a helping hand. I remember how Vasu and Sheela accompanied me to the hospital when I had the surgery even though they were both extremely busy. Vasu, Sheela, Kavi, Kiran and Jay even came over to my place in the evening to help me with the dinner as I lay in bed in pain. They simply did a marvelous job in cheering me up. Madhavi served me breakfast, lunch and dinner on a daily basis and kept calling once in a while to see if I was doing fine. I simply couldn't have asked for more. Similarly, I have no idea what we would have done without the help rendered by Pramod and Vishakha and Chandrasekhar and Meenakshi when Anindita had her bike accident. From getting emergency medicines at 1 in the morning or providing us with restaurant quality food on a daily basis, to providing company to Anindita when she was stuck on her bed, they did all that was possible. Words cannot describe how grateful we both are for the help you gave us when we really needed it. Thank you so much!! Kavitha's daily visits in the evening even in the midst of her busy schedule just to help raise Anindita's spirits were so thoughtful of her. Kiran, thanks a lot for all your help when we moved to our new house. Kavi and Kiran - the magnitude of your warmth, kindness and humility isn't something that one experiences very often. We are so fortunate and privileged to have friends like you.

Holland was far from my first choice to do my PhD studies. It was primarily because I always wanted to live in an English speaking country. But looking

back, I am really thankful that destiny brought me here. The cycling infrastructure of Holland never fails to amaze me. Though I miss the shopping malls back home, the calm and serene environment of our University campus in particular, and Enschede in general, has been a refreshing change. My experiences in Holland, thanks to the people and the place, have been truly *gezellig*.

There is no way this thesis would have come about had it not been for the unwavering support of my family.

I owe nearly everything to my parents for all that I have achieved in life so far. They've strived to provide me with the very best in life. Right since my primary school days, my father has always been the one who encouraged me and made me believe in myself and in everything I did, be it studies, music or even eating (remember Matthew Star?!!) - most of you who know me well, probably know I'm not exactly famous for gobbling up my food in a jiffy (but believe me, things were far worse when I was a kid - I shall spare the reader the details). Actually, it was because of him that I ended up in Twente - after all he'd googled the position online. I was more interested in some other offers I had got, but after some prolonged debates, he finally convinced me to apply to Twente. In retrospect, it was definitely the best decision that I could have made then. Somehow, my father's advice on many crucial decisions during the course of my life have always been bang on the dot even though I didn't consider them so at that point of time. His enthusiasm in everything related to my life is so encouraging. He will always remain a model for me, now that I myself have entered parenthood.

My mother has always been the lighthouse of my life - guiding me along the right path even when things looked bleak. She has sacrificed more than anyone possibly could to make sure that I performed my very best in whatever I pursued. Her ability to provide constructive criticism in areas that are way out of her domain of geomorphology and education always fascinates me and it is because of this, that to this day, I always seek her opinion when it comes to important decisions. If I could ever emulate even half of my mother's capabilities, I would consider myself a successful man. If as some say, reincarnation is true, and if I ever have the chance to choose Babai and Mummy as parents, I would do so again every time!

Rhea, my multi-dimensionally talented sister, though 14 years younger than me, is always a great source of inspiration. She is an exemplary example of brilliance, talent and hard work all packaged into one. I remember there were times when I would get tired of tediously debugging my Matlab simulations codes during the weekends, but seeing my sister slogging at the other end of the

webcam, I would jump straight back to work!

Ever since I came to the Netherlands, my grandmother would always send me letters to find out how I was doing managing things alone. After all, in her eyes, I was always the little kid who would throw up on her sari seconds before boarding the bus that would take me to the kindergarten! I am indebted to Dida for all her warmth and affection. At times, I really miss the presence of my grandfather, who left us when I was 15. Being a professor of English literature himself, it would have been wonderful to have him around to see me complete my PhD. It has been many years since he left us, but I will always remember how proud he felt of his "Dado".

I would also like to thank my in-laws for the regular emails asking how I was progressing. Every email definitely acted as an added incentive to wrap things up as soon as possible! They visited us several times in the last couple of years. Unfortunately, I was hardly able to spend any time with them, as I was tied up with completing my thesis. I really appreciate their understanding and hope that the next time Baba and Ma visit us, we will have some enjoyable times together.

I began life here in Enschede as a bachelor. But now, not only am I married, but I have been blessed with a wonderful baby as well! Anindita joined me here in mid-2005. Looking back, I really can't imagine how this day would have come had it not been for her presence. It is literally impossible to find a more loving and caring wife. No matter how drained or dejected I felt at the end of the day, Anindita's unparalleled eternal exuberance always lifted up my spirits. Be it proof reading every one of my conference papers, whipping up a mouth-watering exotic dish from Bong-Mom's cookbook, soldering battery holders on sensor nodes and deploying them under the curious eyes of kangaroos, organising our highly complicated travel plans following a conference or simply walking with me in the evenings around our lovely campus, Anindita has always supported me in every imaginable way. She's not just my wife but my best friend and I can't thank her enough for simply being the perfect companion one could ever wish for.

Our cute little daughter, Samhita, has been the latest addition to our family and I must say she definitely deserves the most credit for helping me finish off the thesis towards the end. I used to plead to her every night before she was born, not to come out before I submitted my thesis. Well, she's definitely the most understanding baby one can ask for as she decided to make her appearance the very next day after I submitted the thesis to Gerard! And now that she has arrived, she has lighted up a flame of joy in our hearts that will continue to

burn till the very end of time. Thank you so much for being the most wonderful baby we could have asked for!

Yes I do agree that this acknowledgement has turned out to be nearly of epic proportions! But it only goes to show how success is dependent not just on one's own accomplishments but is an amalgamation of all big and small experiences that one encounters in life.

Thank you all once again for being part of this memorable journey that I hope will continue as I set sail to discover more of what life has to offer.

Sincerely,
Supriyo Chatterjea
September 2008,
Enschede,
The Netherlands

DCOSS 2008, Santorini, Greece

Contents

Chapter 1

Introduction

"I think there is a world market for maybe five computers."

Thomas John Watson, Sr.

This is a remark that was made in 1943 by the then president of International Business Machines (IBM), Thomas John Watson, Sr. More recently in 2004, Forrester Research predicted that there would be around 1.3 billion computers worldwide [69] by the year 2010. As Moore's Law, (which states that the number of transistors on a chip doubles every 18 months) has enabled more computing power to be packed in within smaller devices, the basic definition of the computer has seen a phenomenal transformation over the past six decades. This transformation was simply unimaginable in the past - even by those considered to be the stalwarts of the computer industry.

With the advent of further miniaturization of electronics, the past decade has seen the emergence of a new breed of tiny computers known as *wireless sensor nodes* which can be networked together to form *wireless sensor networks* (WSNs). This new generation of tiny computers is anticipated to cause an even greater paradigm shift in the number of computers worldwide. This is because unlike Microsoft's vision of having "a computer on every desk and in every home" for the personal computer [107], sensor networks are envisioned to weave into the very fabric of our daily lives and be present in one form or another in even the most mundane of devices, like in a coffee mug for instance. Initial deployments of sensor networks have already been made in a large variety of applications, e.g. in buildings [142] and bridges [131] for structural monitoring,

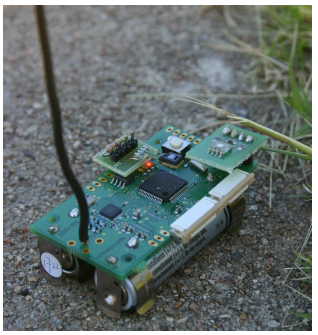

Figure 1.1: The μNode from Ambient Systems

in vineyards [41] and potato fields [87] to improve harvests, in unmanned aerial vehicles (UAVs) for military applications [119] and even in the depths of the ocean for environmental monitoring [47].

This fast-growing array of applications will result in an explosion in the number of devices which are capable of communicating with one another. Never since the birth of computer science has the research community faced a problem of this magnitude - making thousands or perhaps even millions of devices communicate seamlessly with each other without any external human intervention whatsoever. This makes it imperative to devise new and novel solutions.

While the general line of research in the development of conventional computers has been to find ways to squeeze more computation power into a smaller form factor, WSN research may be considered to have taken a step backward as sensor nodes typically use relatively primitive hardware that are more reminiscent of the early 1980s. Present day wireless sensor nodes [34, 58, 109], like the ones shown in Figure 1.1, typically have processors which run at around 8-16MHz and only have a few kilobytes of RAM, e.g. 4-10kB. However, what differentiates WSN research from the development of conventional computers are the motivating factors - energy-efficiency and self-organizing operation instead of increased computing power. The reason for this is that sensor nodes are typically battery-powered devices and should be able to operate unattended for several years.

1.1 Focus and approach of research

We first provide the reader with a macro perspective of some of the major areas of research in the field of sensor networks. Since a significant proportion of our work deals with the networking aspects of sensor networks, we go on to elaborate how our research approach differs from the approach taken when designing networking protocols for conventional computer networks. We finally describe the specific areas we have focused on in this thesis.

1.1.1 Primary areas of WSN research

The broad spectrum of WSN research may be split into several categories. Below, we briefly describe each category.

1. *Sensing*: Research in this area involves a few topics such as the sensors themselves, packaging technologies, the sensor platform, transceiver, antenna and power generation.

   The sensors can vary from simple temperature or humidity sensors to more complicated varieties which measure turbidity or dissolved oxygen. As sensor networks are generally meant to be deployed in high spatial densities, for long periods of time and in possibly harsh environments, the majority of research in this domain concentrates on minimizing production cost and power consumption and increasing durability of the sensors, e.g. by using sophisticated packaging technologies. Before the birth of sensor networks, the spatial granularity of monitoring was generally performed at very coarse resolutions. Users were expected to manually recalibrate the sensors on a periodic basis to ensure the integrity of all the collected data. Thus apart from simply focusing on hardware, distributed algorithms are also being developed to enable sensors to calibrate themselves autonomously [42]. More recently, a new class of low-power "camera sensor networks" has emerged. Thus instead of simply transmitting numerical sensor data, these devices actually transmit low-resolution image data [86].

   The sensor platform generally consists of a micro-controller with on-chip RAM and Flash, a transceiver, a digital and analogue I/O interface for connecting a wide variety of sensors and actuators and in many instances an in-built antenna. Most of the existing platforms have been created using off-the-shelf components. Apart from using the right combination of low-power components, emphasis is also placed on the casing that contains the sensor node. The casings need to be both weather and shock

proof to ensure that the nodes are able to survive extended periods in harsh environmental conditions [47]. Additionally, in order to improve communication reliability and efficiency, a significant amount of research is being conducted in the fields of low-power transceivers [127] and antenna design [79, 104].

While present day sensor nodes are typically battery-powered, a significant amount of research is being dedicated to energy-harvesting [120, 123]. Nodes could make use of the very environment they are deployed in to generate power, e.g. solar energy, vibration, wind, etc. Although these techniques may not generate huge amounts of energy, they are still useful, as compared to batteries, they provide sensor networks with a nearly infinite source of energy.

2. *Communication protocols*: Communication protocols ensure that sensor nodes distributed over a wide area are able to reliably communicate with one another wirelessly in an energy-efficient manner. These algorithms address issues such as Medium Access Control (MAC), routing, reliable transport, time synchronization, clustering, etc. Depending on the application, these algorithms may also need to support mobility.

   Having a large-scale, high-density sensor network means that a lot of nodes will have to share the wireless channel to transmit data. If two nodes that are within transmission range decide to transmit a message to each other at precisely the same time, both messages will get corrupted. The MAC protocol ensures that message transmissions between nodes are scheduled in a way such that message corruptions do not occur [77, 88]. MAC protocol research for WSNs generally focuses on striking the optimal balance between energy-savings and latency and also deals with techniques that ensure that nodes can automatically choose new schedules if any changes in the network are detected (e.g. a node dies or a new node is added).

   In conventional computer networks or wireless ad-hoc networks, queries are usually routed from a source to a specific destination (or node) which may be identified by an IP address. As users of WSNs are typically more interested in the actual data that is being collected by a *subset* of the deployed sensor nodes, rather than observing the behaviour of a *specific* node, queries are routed to the set of nodes which are sensing and thus producing the required data. In this approach, when a node receives a query, it forwards it to another neighbour, not based on the *ID* of a particular destination node but based on the *data* that is requested

by the user. This is known as data-centric routing [82]. Also, rather than performing end-to-end routing, WSNs usually rely on neighbour-to-neighbour routing.

While traditional data collection from source to sink nodes may be tolerant to message loss, data dissemination from sink to source nodes (e.g. for reprogramming nodes) is highly sensitive to message loss. For such applications, it is essential to have reliable transport protocols [100, 101]. Note that a *source node* refers to a node that produces the data by sampling its sensors and a *sink node* refers to the node that requests the data, i.e. it is the final destination of the data collected by nodes in the network.

Distributed protocols for sensor networks may be heavily dependent on time. Additionally, many applications require sensed data to be time-stamped before being transmitted to the sink. As clock drift may be a common phenomenon in low cost sensor nodes, it is essential to have distributed time synchronization protocols that ensure that the clock drift is kept within limits that would meet the requirements of the end-user and communication protocols [66].

3. *Operating systems and reprogramming*: A sensor node is not simply a device that transmits sensed data. A present-day sensor node typically has a host of resources: the processing unit, volatile and non-volatile memory, interface resources such as DACs, ADCs, UARTs, interrupt controllers, counters, the transceiver, and the various sensing devices. Managing all these resources in an energy-efficient manner in a highly resource-constrained environment requires the presence of a lightweight operating system [65].

   Efforts are also being made to ensure that the operating system is able to adapt to the changing requirements, e.g. rather than being hard-coded with a single routing protocol, a node may need to use different protocols at different times in order to continue operating efficiently in any dynamic environment. This requires nodes to be reprogrammable [68].

4. *Distributed services*: The computation power of individual sensor nodes enables them to run a host of services locally. These services can then operate in a distributed manner due to the wireless communication capability of the nodes. Some examples of distributed services are localization [65], data aggregation, query processing, distributed data storage and management [45], event/outlier detection and distributed actuation and

control [152]. With the exception of localization, the remaining examples deal with the data that is generated within the network.

In most applications, readings from a sensor node would probably be deemed useless if the user is not aware of the location of the node. While the Global Positioning System (GPS) is able to provide location information, it is unsuitable for sensor networks because of its high cost and power consumption. A GPS device also cannot work in an indoor environment. Current localization algorithms for WSNs generally use information such as connectivity, radio signal strength information (RSSI) or sound intensity. The main issue here is to investigate algorithms that result in the most precise location estimation without using any additional hardware in the most energy efficient manner.

Managing the data generated by the WSN within the network itself, is a process that is particularly unique to WSNs. In conventional networks, intermediate nodes (i.e. nodes that lie between the source and destination) simply act as relaying nodes. These relay nodes do not read, analyse or act upon any of the data they are forwarding. The sensor network designer, however, takes advantage of the computation capabilities of a node and instructs the data management layer of an intermediate node to analyse the actual data passing through it and act upon it if necessary. This is a strategy that is used to improve energy efficiency. As as example, an intermediate node may refrain from transmitting duplicate data messages. The data management component may also handle issues such as deciding when and where to perform data fusion, how to evaluate queries that a node has received and how and where to store acquired data within the network if required. Additionally, depending on the application requirements, it might also help identify events or outliers, or help perform distributed actuation.

5. *Security*: Security plays an essential role in WSNs that are deployed in sensitive locations. WSN security can be classified into information security and operational security [89]. Information security should ensure that confidential information is never disclosed and the integrity and and authenticity of information is always guaranteed. Operational security should ensure that the network continues to function even if some of its components are attacked. Issues such as limited computational power and memory and the open wireless medium make WSN security a non-trivial issue.

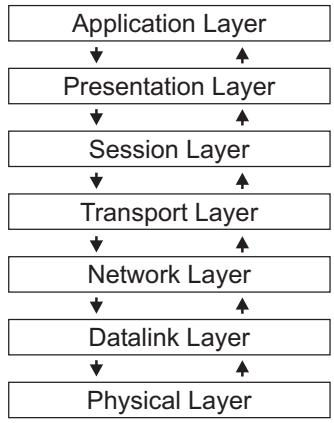

Figure 1.2: The OSI Model

6. *Testing and evaluation*: Since WSNs are large-scale and complex systems, it is essential to test any developed protocol or algorithm through simulations, emulations, prototyping and real large-scale deployments. Simulators and emulators help evaluate the entire system or even a subset of the developed components [143, 90]. Simulators are used to not only simulate the performance of protocols but are also used to simulate the environment where a sensor network may be deployed [83, 149]. Deployment and debugging tools are required to help deploy WSNs and evaluate their performance in real-life [125].

1.1.2 Research approach

When developing algorithms for computer communication networks, i.e. networking protocols, it is common practice to specify which layer of the OSI (Open Systems Interconnection) [153] model the algorithm has been designed for. The layered approach of the OSI model ensures that every layer operates completely independently (Figure 1.2). Thus the operations performed within a particular layer are not dictated by what happens within any other higher or lower layers. This is especially advantageous for vendors creating software for personal computers (PCs) as PCs are general purpose machines that are designed to be used for a wide range of applications. For example a particular model of a desktop PC may be used in an organization primarily for performing Matlab simulations

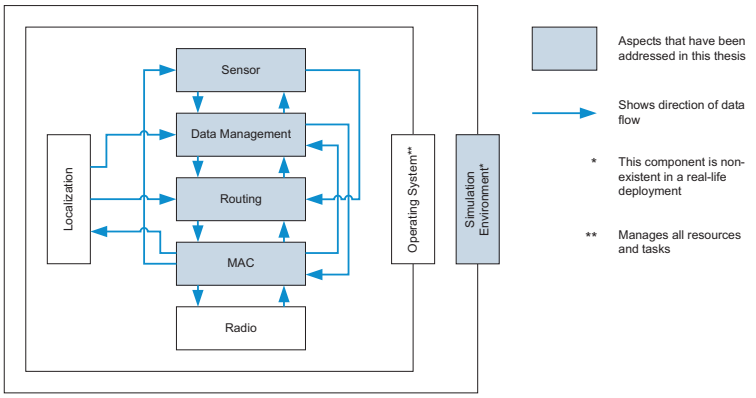

Figure 1.3: Architecture of a wireless sensor node

while the same model may be used by a home user for mainly video conferencing and playing games. As another example, a vendor designing a web browser will not have to worry about the interface used to connect to the Internet. Following the OSI model will ensure that the browser works whether the connection is via Ethernet or Wireless LAN. In fact, the economic viability of a product may be questionable if it is not designed following this modular approach as the company would have to provide complete solutions and inter-operability between different vendors would be nearly impossible.

The situation for WSNs, however, is vastly different as WSNs are typically *application-specific* networks. Thus the precise reason for deploying a particular WSN would be known prior to deployment. Instead of placing emphasis on inter-operability, the focus here is on energy-efficiency and ensuring that the quality of the acquired data meets the user's requirements. Thus the main philosophy is to maximise information usage. As an example, if there is a piece of information that is being used by a particular layer within a node, the objective is to find out which other layers can utilise this same information in order to reduce redundancy and improve energy-efficiency. We now illustrate how we have used this philosophy for our design approach.

The data flow diagram in Figure 1.3 illustrates how the various components in our sensor network architecture exchange information with each other. It is immediately apparent that the level of inter-dependence between the various components is much greater than the layered approach used in conventional computer communication networks. For example, the localization component

may estimate the location of a node by using connectivity information provided by the MAC layer. This location estimate may then be used by the routing component to perform *geographic routing*.

1.1.3 Focus of this thesis

This thesis primarily focuses on the following issue:

How can data be extracted from
a wireless sensor network
in an efficient manner?

The techniques we suggest can be used for two different classes of applications:

1. Applications that only require a subset of all the data in the network, provided certain conditions are satisfied at some specified time. Data for such applications are extracted using *range queries*. These queries either provide the user with a snap-shot of the various physical parameters being monitored or provide streaming data for a short duration.

2. Applications which require all the data (i.e. *raw data*) to be gathered from all the sensors in the network at periodic intervals. Data for such applications are extracted using *long-running queries*.

We provide examples of the above-mentioned classes of applications in Section 2.3.

Our hypothesis is that regardless of the class of applications being addressed, a cross-layered approach would help develop solutions that are efficient, adaptive and self-organizing.

For the first class of applications, we present a framework that ensures that one-shot range queries are routed only to the relevant regions of the network instead of carrying out flooding. The same framework is also used to assign an appropriate amount of bandwidth to regions that are expected to generate more data with respect to the incoming query. This allows precious energy-resources to be spent only where gains are expected. This part of the work covers the following components shown in Figure 1.3: Data Management, Routing and MAC.

For the second class of applications, we have designed two algorithms that help extract raw data in an energy-efficient manner. The first algorithm, that takes advantage of spatial correlations that may exist between the readings of neighbouring sensor nodes, is a scheduling algorithm which decides when a particular node should aggregate data. The second algorithm helps save energy by sampling the sensors in an energy-efficient manner by taking advantage of temporal correlations that may exist between successive sensor readings. We have also devised a method to generate large spatially and temporally correlated synthetic datasets that are suitable for testing algorithms for WSNs. This part of the work covers the following components shown in Figure 1.3: Data Management, Sensor and Simulation Environment.

1.2 Contributions

The major contributions in this thesis are five-fold:

- *Contribution 1*: We **provide a taxonomy of distributed data management techniques** that are used in WSNs. The state-of-the-art in both energy-efficient query dissemination and data acquisition are described. We highlight the advantages and disadvantages of the various techniques and this in turn helps us define the reasons why we have pursued the areas of research that are contained in this thesis.

- *Contribution 2*: We **present an energy-efficient algorithm for servicing one-shot range queries**. When disseminating one-shot queries, they can either be flooded to the entire network or be directed only towards the areas which are expected to return the required data. Performing directed query dissemination would naturally result in energy-savings. Our directed query routing strategy routes queries based on both *static and dynamic* attributes. In a static network, location would be an example of a static attribute while temperature would be an example of a dynamic attribute. Thus as an example, a particular one-shot range query may be directed to different parts of a network at different times of the day due to changing temperature gradients. In order to direct the queries accurately, the routing tables need to be kept updated. However, performing excessive updates would cause the cost of directed query dissemination to exceed that of flooding. Thus the frequency of transmission of updates is computed based on the variation of the measured parameter and the number of queries injected into the network. Steps are taken to ensure

that the total cost of the whole mechanism is always kept below that of tree-based flooding.

- *Contribution 3*: We **present a framework that allows a MAC protocol to adapt its operation based on the requirements of the application**. MAC protocols usually allocate the same amount of bandwidth to all nodes within the network. The problem with this strategy is that not all nodes may require the same amount of bandwidth, e.g. there may be times when some parts of the network may generate more data than other parts due to the requirements stated within a particular range query injected into the network. Our framework allows individual nodes to adapt their operation of the MAC to cope with such variations in data traffic rates using only locally available information.

- *Contribution 4*: We **present a scheduling algorithm that is used to extract data from a WSN in an energy-efficient manner**. The purpose of the algorithm is to decide when a particular node should be in charge of aggregating data. Nodes are able to choose schedules in a distributed manner and can adapt autonomously to topology changes by using cross-layer information provided by the underlying MAC protocol. The algorithm is shown to have self-stabilizing properties. Both our simulation and implementation (on actual sensor nodes) results illustrate an 80% reduction in the number of message transmissions.

- *Contribution 5*: We **present an adaptive algorithm for controlling the sensor sampling frequency**. The main sources of energy consumption in a sensor node are the radio transceiver and the attached sensors. This algorithm helps reduce the sampling frequency of the sensors by predicting sensor readings using *time-series forecasting* rather than actually sampling the sensors. When a measured parameter varies gradually in a predictable manner, the sampling frequency is reduced. Similarly, the frequency is increased when the measured parameter varies rapidly and is unpredictable. Also the sampling frequency used by a node is dependent on the number of neighbours it has, i.e. the larger the number of neighbours a node has, the lower the sampling rate. Thus every node can autonomously change its sampling frequency if a change in the local topology is detected.

1.3 Structure of thesis

In the next chapter we provide the reader with an overview of sensor networks, what they are, how they operate and where they are used. Chapter 3 provides a taxonomy of the distributed data management techniques that are used in WSNs and lays the foundations for the work presented in Chapters 4-6 (Contribution 1). Chapter 4 illustrates how one-shot range queries can be managed in an energy-efficient manner by performing directed query dissemination and adapting the MAC based on the requirements of the application (Contributions 2 and 3). Chapter 5 describes the scheduling that is used to extract data in an energy-efficient manner (Contribution 4). Chapter 6 presents material on energy-efficient sensor sampling and synthetic data generation (Contribution 5). We conclude this thesis in Chapter 7 summarizing the key results and highlighting the open research areas that still need to be investigated.

1.4 Selected list of publications

Below is a selected list of publications that form the core of this thesis. We refer the reader to the bibliography for a complete list of publications that were made during the course of this research.

Journals

1. 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. In *ACM Transactions on Sensor Networks*, To appear. (*Contributes to Chapter 5.*)

2. S. Chatterjea and P. Havinga. A Taxonomy of Distributed Query Management Techniques for Wireless Sensor Networks. In *International Journal of Communication Systems*, 20 (7). pp. 889-908, Wiley, 2006. (*Contributes to Chapter 3.*)

3. S. Chatterjea, L.F.W. van Hoesel and P. Havinga. A Framework for a Distributed and Adaptive Query Processing Engine for Wireless Sensor Networks. In *Transactions of the Society of Instrument and Control Engineers*, E-S-1 (1). pp. 58-67, 2006. (*Contributes to Chapter 4.*)

Book chapters

1. S. Dulman, S. Chatterjea, T. Hoffmeijer, P. Havinga, and J. Hurink. Architectures for Wireless Sensor Networks. In *Embedded Systems Handbook, (R. Zurawski ed.)*, CRC Press, August 2005. (*Contributes to Chapter 3.*)

Conferences and workshops

1. S. Chatterjea, and P. Havinga. An Adaptive and Autonomous Sensor Sampling Frequency Control Scheme for Energy-Efficient Data Acquisition in Wireless Sensor Networks. In *Proceedings of the Fourth IEEE Conference on Distributed Computing in Sensor Systems* (DCOSS), June 2008, Santorini, Greece. Pages to appear. Lecture Notes in Computer Science. Springer Verlag. (*Contributes to Chapter 6.*)

2. S. Chatterjea, T. Nieberg, Y. Zhang and P. Havinga. Energy-Efficient Data Acquisition using a Distributed and Self-organizing Scheduling Algorithm for Wireless Sensor Networks. In *Proceedings of the Third IEEE Conference on Distributed Computing in Sensor Systems* (DCOSS), June 2007, Santa Fe, USA. pp. 368-385. Lecture Notes in Computer Science 4549 (LNCS4549). Springer Verlag. (*Contributes to Chapter 5.*)

3. S. Chatterjea, S. de Luigi and P. Havinga. An Adaptive, Directed Query Dissemination Scheme for Wireless Sensor Networks. In *Proceedings of the Thirty-Fifth IEEE Conference on Parallel Processing Workshops* (ICPPW), August 2006, Columbus, USA. pp. 181-188, IEEE Computer Society Press. (*Contributes to Chapter 4.*)

4. S. Chatterjea, L.F.W. van Hoesel and P. Havinga. AI-LMAC: An adaptive, information-centric and lightweight MAC protocol for wireless sensor networks. In *Proceedings of the First IEEE Conference on Intelligent Sensors, Sensor Networks and Information Processing* (ISSNIP), December 2004, Melbourne, Australia, IEEE Computer Society Press. (*Contributes to Chapter 4.*)

Chapter 2

Background

This chapter introduces the reader to the field of wireless sensor networks. After presenting the functions of the various building blocks of a sensor node platform, we describe a wide spectrum of applications to illustrate how this technology is making its presence felt in everyday life. We then give a detailed description of three applications that are particularly relevant to this thesis as they have provided the motivating factors behind the assumptions and design decisions we have made in the forthcoming chapters. Based on the requirements posed by sensor network applications, we then highlight the main characteristics that should be present in any protocol designed for sensor networks. As we strongly advocate using a cross-layered approach to designing sensor network protocols, we first give an overview of the MAC protocol our work is based on and conclude by illustrating the benefits that can be acquired by allowing various components of the sensor node architecture to extract information from the underlying MAC layer.

A common misconception is that a wireless sensor node is simply a sensor with an attached radio transmitter. However, there are two main differences. Firstly, the radio of a sensor node is able to both transmit and receive, i.e. it is a transceiver. Secondly, in addition to the attached sensors and the transceiver, the sensor node has a limited amount of built-in "intelligence" due to its microcontroller with on-chip RAM and Flash. It is this intelligence that allows a sensor node to operate autonomously and in an energy-efficient manner.

The design of the architecture of the sensor node platform is not the only aspect that revolves around energy-efficiency. The mode of communication used in WSNs is also designed to minimise power consumption. Let us first take a look at how some other wireless devices operate: the mobile phone and a wireless notebook computer. In order to connect to their respective networks, both these devices communicate directly with the base station essentially forming a single-hop network. However, it should be noted that this is not a very energy-efficient form of communication as according to the Friss Transmission Equation [7], the energy-cost of radio transmission is directly proportional to the square of the transmission distance. While mobile phones and wireless notebooks are supposed to be energy-efficient, they are not expected to work for years without recharging. As an example, a typical WLAN PCard can use 1.425W during the transmit operation while the sensor node shown in Figure 1.1 would use only 16mW. This makes it essential for WSNs to use other more efficient forms of communication. In order to reduce power consumption, WSNs use *multihop communication* as it eliminates the need for long range transmission. Multihop communication allows a message generated by a source node to be propagated to the destination node with the help of intermediate nodes, which forward the data in a hop-by-hop fashion. Multihop communication also enables networks to be deployed over larger distances without increasing the power consumption due to radio communication.

Traditionally many monitoring applications have been using data loggers [118, 114, 115] to measure various physical parameters such as temperature and humidity. As data loggers are generally very expensive, most users of such systems can only afford to use a small number of these devices for their monitoring applications thus making them impractical for fine-grained monitoring. Moreover, the malfunction of one data logger would result in a complete loss of data for the entire area the data logger is supposed to cover.

WSNs help eliminate many of the problems posed by such data loggers. Firstly, since they are low in cost (the price is projected to be around $5 in the future [71]) users should be able to afford deploying large, high density WSNs. Such densely packed deployments will allow users to collect data at

unprecedented fine-grained spatio-temporal resolutions that are unachievable using conventional data loggers. The redundancy caused by such high density deployments also ensures that the network continues to perform its task even if a small proportion of the nodes fail. Another feature unique to WSNs is the fact that instead of working individually, sensor nodes often collaborate with one another to reduce power consumption and improve the quality of the data collected.

In the following sections we first give a brief overview of the WSN platforms that are currently available in the market. The next section provides a general list of WSN applications that have already been deployed or are intended to be deployed in the near future and describes a few of the applications relevant to this thesis in greater detail. We then state the general requirements of protocols designed for WSNs based on the foundations laid by the WSN applications described earlier. As mentioned earlier, we have emphasized on taking the cross-layered approach to designing WSN protocols through out this thesis. This implies that much of our work is dependent on the operation of the underlying MAC protocol. In this regard, the final section of this chapter first provides the reader with a brief overview of the MAC protocol we have based most of our work on in this thesis. We also describe the benefits obtained by using the cross-layer information provided by the chosen MAC protocol.

2.1 WSN platforms

Sensor node platforms typically consist of a host of sensors, a microcontroller, a transceiver and a power supply. Table 2.1 provides a summary of the various platforms currently available in the market and in academia. While the specifications of the various components may vary from vendor to vendor a few important characteristics of the hardware can easily be identified. All the platforms have a slow processor, minimal storage, low bandwidth and a scarce energy supply. We now give a brief overview of each of the components forming the sensor node.

2.1.1 Sensors

Sensors are responsible for measuring the physical environment. There are a wide variety of sensors available in the market that can be interfaced with sensor nodes: ambient temperature, relative humidity, solar radiation, light, acceleration, magnetic field, voltage, current(DC and AC), sound, ultrasound,

Sensor node platform	Microcontroller				Transceiver		
	CPU	Clock freq (MHz)	RAM (kB)	Program memory (kB)	Type	Freq (MHz)	Max data rate (kbps)
weC [19]	Atmel AT90S8535	4	0.5	8	RFM TR1000	916.5	10
Rene mote [19]	Atmel AT90S8535	4	0.5	8	RFM TR1000	916.5	10
Rene2 mote [19]	Atmel ATmega163	4	1	16	RFM TR1000	916.5	10
MIT μAMPS [108]	Intel StrongARM SA-1100	206	16384	512	National Semiconductor LMX3162	2400	1024
Crossbow MICA [6]	Atmel ATmega128L	4	4	128	RFM TR1000	433, 915	40
Crossbow MICA2DOT [6]	Atmel ATmega128L	4	4	128	Chipcon CC1000	315, 433, 915	38.4
Crossbow MICA2 [6]	Atmel ATmega128L	7.37	4	128	Chipcon CC1000	315, 433, 915	38.4
Crossbow MICAz [6]	Atmel ATmega128L	4	4	128	Chipcon CC1000	2400	250
EYES (Nedap) [137]	TI MP430F149	2	8	60	RFM TR1001	868.35	57.6
EYES (Infinion) [73]	TI MP430F149	2	8	60	Infinion TDA 5250	868-870	64
BTnode rev3 [4]	Atmel ATmega128L	7.37	244	128	Chipcon CC1000	433, 915	38.4
Moteiv Tmote Sky [109]	TI MP430F149	8	10	48	Chipcon CC2420	2400	250
Ambient μNode [2]	TI MP430	4.6	10	48	Nordic nRF9E5	868, 915	50
Ambient SmartTag [2]	8051	16	-	4	Nordic nRF9E5	868, 915	50

Table 2.1: A list of various sensor platforms

barometric pressure and even rainfall. The nature of these sensors not only affects the cost and physical size of the sensor nodes but also can affect the lifetime of the WSN. As can be seen from Table 2.2, certain sensors can consume significant amounts of energy. This is the reason why apart from simply trying to reduce message transmissions, we discuss techniques that may be used to reduce the number of sensor samples acquired (Chapter 6).

2.1.2 Microcontroller

The microcontroller together with its on-board RAM, Flash and/or EEPROM is the main contributing factor to the "intelligence" of a node. It usually has a low-power 8-bit or 16-bit RISC core and generally supports several sleep modes and operates at a maximum frequency of just a few MHz. The low frequency is

Sensor type	Current (mA)	Energy Per Sample (mJ)
Solar radiation [18]	0.350	0.525
Barometric pressure [11]	0.025	0.003
Humidity [17]	0.500	0.500
Surface temperature [13]	5.6	0.0056
Ambient temperature [13]	5.6	0.0056
Accelerometer [3]	0.6	0.0048
Magnetometer [9]	5	0.2595

Table 2.2: Power consumption of various types of sensors [95]

not considered to be a very significant detrimental factor as nodes are supposed to process data in unison rather than individually, i.e. they should be sharing responsibility. However, there are times when certain algorithms may have to be optimised when dealing with large datasets in order to shorten processing times [102]. Table 2.1 shows the microcontrollers used on the various sensor node platforms.

2.1.3 Radio

In many applications, the radio consumes the most energy among all the components in the sensor platform - thus the majority of algorithms deal with minimising usage of the transceiver as much as possible. The transceiver usually has a single channel, a low data rate and operates at the unlicensed bands at 868MHz, 902MHz and 2.4MHz depending on which country the product is designed for. Note that the cost of transmitting one bit of data can be as much as executing 1000 CPU cycles [96]. This is the main reason why WSN research focuses mainly on processing data within the network rather than transmitting it.

2.1.4 Power source

The power source is probably the main reason why sensor network research first began. If there were no constraints on the amount of energy reserves, there would be no need to have energy-efficient architectures and communication protocols. As an example, in a real-life experiment conducted by Cardell-Oliver et

al. [43], using MICA2 motes running on Alkaline, NiMH and LiSO2 batteries lasted for approximately 2, 8 and 28 days respectively. Additionally, battery capacity has little more that doubled in 10 years from 280 Whr/l (watthours per liter) in 1995 to 580 Whr/l in 2005 [55]. This is in stark contrast to the complexities of ICs which generally doubles every 18 months. Such facts exacerbate the need to to develop highly efficient communication and data management protocols for WSNs.

Regardless of how energy-efficient WSNs protocols are and how technologically advanced battery technology is, a battery by itself will always represent a finite source of energy. Thus an alternative would be to use *energy harvesting* techniques. Energy harvesting enables a node to extract power from the very environment it is monitoring. Some existing techniques in the literature can be classified into four main categories:

- *Harvesting vibrational energy*: Meninger et al. describe a microelectromechanical (MEMS) device that is capable of converting ambient mechanical vibration into electrical energy [105]. The device transduces energy by means of a variable capacitor and is capable of producing up to 8μW of power.

- *Harvesting light energy*: Solar cells have been used in the PicoRadio project to power PicoNodes. However, form factor may be a problem as one square centimeter can only contribute around 0.15mW (cloudy day) to 15mW (direct sunlight) of power [122].

- *Harvesting thermal energy*: Researchers are also investigating methods that take advantage of thermal gradients. The general idea is to make use of the Seebeck effect which explains the existence of a voltage between two ends of a metal bar when a temperature difference of ΔT exists in the bar [21].

- *Harvesting wireless power*: Powercast has [16] deployed a WSN using sensor nodes that use rechargeable batteries at the Pittsburgh Zoo and PPG Aquarium [10]. A Powercast transmitter is plugged into a permanent power supply. This transmitter sends out a continuous, low RF signal to all the nodes in range. The sensor nodes are equipped with a Powercast receiver, which is the size of a fingernail and the harvested power is used to continuously charge the batteries.

It is important to keep in mind however, that all the techniques mentioned above are only able to generate extremely small amounts of electricity. Thus

energy-efficient communication and data management protocols, such as the ones described in this thesis will still be very relevant.

2.2 Applications

In order to illustrate how significant the impact of WSNs is projected to be in the near future, we first provide the reader with a brief list of applications areas where WSNs *are* or *can* be deployed. We then describe some specific applications that are particularly relevant to the work presented in this thesis.

- *Military & security*: One of the first WSN research initiatives was funded by the Defense Advanced Research Projects Agency (DARPA) under the SensIT program in 1999. In an experiment funded by SensIT, at the Marine Corps Air/Ground Combat Center in Twentynine Palms, California, researchers from UC Berkeley used an unmanned aerial vehicle (UAV) to (i) drop sensor nodes onto a road to track vehicles on the ground, (ii) collect the tracking information from the network and (iii) transfer the collected information back to the base camp [1]. Apart from tracking vehicles, WSNs can also be used to pinpoint the location of a sniper. The authors in [129] describe a system which uses sensor nodes to detect the difference in arrival times of acoustic signals when a gun is fired. The field trials showed 3D localization accuracy of 1.3 meters and latency of less than 2 seconds. Remote border security is another application area where WSNs can be used to reduce manpower cost and improve the level of security. Instead of having to place security forces to monitor any possible intrusions, WSN systems have been developed to detect such events remotely [145, 136].

- *Intelligent buildings*: With the skyrocketing demands for electricity, efforts are currently underway to design buildings that would minimise the costs of *space conditioning*. Space conditioning refers to the artificial cooling or heating of any space and is the dominant consumer of energy in the commercial-building arena [121]. Rabaey et al. [121] claim that the use of WSNs would reduce energy consumption by 44% due to space conditioning. The ultimate purpose of intelligent buildings is not only to save energy by using it more efficiently but also to allow buildings to carry out their functions in a better and more efficient manner. For example, the same WSN infrastructure used to carry out space conditioning may be

used for informing the emergency services of the presence of a fire or for the operation of the intelligent light control system [130].

- *Precision agriculture*: WSNs may be employed to ease the tasks of the modern-day farmer by continuously monitoring parameters such as soil temperature, moisture and salinity. This would allow the farmer to act immediately to conditions that are adverse to the growth of crops and thus ensure a higher yield. Experiences in deploying a real-life network for monitoring a potato field are described in [87]. WSN applications in agriculture are not just limited to growing crops. They are used to monitor animals as well. The Hogthrob project [8] aims to attach sensor nodes to sows. Currently sows are tagged. However, the farmer needs physical contact with the tags to identify the animal. Furthermore, farmers need to manually detect when a sow enters its heat period. Sensor nodes with built-in accelerometers would allow farmers to track the sows remotely and would also help identify when a sow is in heat. This is because it is known that a correlation exists between the movement of sows and its heat period.

- *Asset tracking/monitoring*: Many companies use Radio Frequency Identification (RFID) tags for monitoring their assets. However, RFID tags have their contraints, e.g. once a tag is outside the range of the RFID reader, the user would not be able to trace the tag. RFID tags also cannot provide the user with any information about the conditions surrounding the tag. WSNs, however, can not only perform the tasks of RFIDs but can do a lot more. For example, in the Cobis project [5], British Petroleum uses WSNs to continuously monitor containers containing hazardous chemicals. The WSNs not only allow users to track the location of the individual containers but can also help raise the appropriate alarm if (i) the number of containers stored in a location exceeds the maximum allowable storage limit, (ii) the goods are stored in an area where the environmental conditions do not suit the safety requirements, (iii) a container is stored next to another container containing an incompatible substance. This clearly demonstrates how sensor nodes can use their built-in "intelligence" to make autonomous decisions.

- *Environmental monitoring*: Large-scale environmental monitoring has usually been performed using remote sensing. Remote sensing techniques have been traditionally relegated to the implementation of airborne or spaceborne configurations, e.g. using satellites. However, satellites do not al-

low continuous monitoring of a particular location. The quality of images taken by satellites are also dependent on weather conditions (i.e. cloud cover). WSNs can be used to complement remote sensing data from satellites as they can be used to observe environments at high spatio-temporal resolutions [36].

- *Disaster prediction & management*: The ability to predict impending disasters could have a major impact on saving lives. The continuous monitoring capability of WSNs, could help with making such predictions. For example, sensor nodes equipped with seismoacoustic sensors have been deployed in northern Ecuador to monitor the behaviour of active volcanoes [144]. The data collected from such networks could be used to predict life-threatening eruptions. WSNs are also being used to complement deep-water tsunami detection buoys to help improve the detection of tsunamis [25]. In addition to predicting disasters, WSNs can also be used in the management of disasters after they occur. Researchers in the RescueNet project intend to embed sensor nodes into the structures of buildings [23]. These nodes remain dormant until a collapse is detected. They then collaborate to help create a 3D view of the collapsed interior identifying the structurally strong support walls and cavities. The WSN is also able to locate survivors using heartbeat sensors. The collected information is then propagated to the rescue team using energy-efficient routing algorithms.

- *Transportation*: With a rapid increase in the number of vehicles on the road worldwide, intelligent road management systems are seen as the key to managing transport infrastructure efficiently. Researchers from Berkeley describe a WSN that uses magnetometers to detect the presence of vehicles, speed and also detect the vehicle type, e.g. passenger, van, SUV, etc. The prototype is able to identify vehicles with 97% accuracy [53]. The information provided by the traffic monitoring system can subsequently be used to optimise traffic light settings at urban intersections. Road users can also use this information to better plan their activities and adjust their routes.

- *Health*: The percentage of older people in developed countries is fast increasing. As there may be a large shortage of qualified care givers, WSNs are expected to play a crucial role in the medical sector by continuously monitoring the vital signs of the elderly. For example, the CAALYX [26] project aims to develop a device that can be worn by the elderly. The

device will be able to continuously sense vital signs of life, detect falls, and communicate automatically, in real-time with the patients' care-giver in case of an emergency.

2.3 Applications relevant to this thesis

In this section, we describe three applications that are of particular relevance to this thesis. The first two applications in sections 2.3.1 and 2.3.2 are related to the work presented in chapters 5, and 6 and focus specifically on the user injecting long-running queries. The third application in section 2.3.3 is related to the work presented in chapter 4 and focuses on the user injecting snap-shot queries.

2.3.1 Sensor networking the Great Barrier Reef

The Great Barrier Reef (GBR), located along the north-east coast of Australia is made up of over 3,200 reefs and extends over an area of 280,000km^2. It is the world's largest coral reef system. The GBR however, is under threat due to the following factors:

- *Global warming*: A temperature rise of between 2 and 3 degrees celsius would result in 97% of the Great Barrier Reef being bleached every year [84].
- *Pollution*: Rivers flowing into the GBR in the north-east coast of Australia flow through large areas of farmland. Thus excess fertilisers and pesticides flow from these farmlands into the GBR.
- *Overfishing*: It is quite common for fishermen to catch unwanted species of fish, e.g. dolphins and turtles. This can cause major disruptions to the food chain and thus eventually harm the corals in the GBR.

The impact of these threats is clearly visible as they cause *coral bleaching* and also adversely affect the eco-system. Coral bleaching is a stress condition that causes the breakdown of the symbiotic relationship between the corals and unicellular algae known as *zooxanthellae*. It is these microscopic plants that provide a coral with its normal healthy colour. When coral bleaching occurs, the algae are expelled from the coral tissue resulting in the corals becoming white. This is illustrated in Figure 2.1. While the corals do not initially die and

Figure 2.1: The effects of coral bleaching (Adapted from [28])

can recover from coral bleaching, prolonged periods of stress can result in the eventual death of a coral.

In order to ensure the long-time survivability of the GBR, it is essential to understand the precise impact that global warming, pollution and over-fishing play in the destruction of the GBR. While global warming is a cause that cannot be readily controlled, pollution and overfishing are. Being able to monitor environmental parameters such as temperature, light, salinity, level of pollutants, etc. at real-time and at a high spatial and temporal resolution would enable scientists to better understand the underlying complex environmental processes that help shape the behaviour of the biological and physical characteristics of the GBR. As an example, if high levels of pollutants (e.g. pesticides) are detected in the GBR, farmers along the coast can be advised to reduce the amount of pesticides that are used. Also, monitoring temperature profiles and fish species can help identify the kind of temperatures preferred by certain fish species. This information can then be used by fishermen to help them track down the fish they are interested in instead of capturing large numbers of unwanted fish species.

We are currently working together with the Australian Institute of Marine Science (AIMS) [32] to set up a large-scale wireless sensor network to monitor various environmental parameters on the Great Barrier Reef (GBR) in Australia. Scientists at AIMS intend to use the collected data to study coral bleaching, reef-wide temperature fluctuations, impact of temperature on aquatic life and pollution.

One of the reefs under study is the Davies Reef which is approximately 80km north-east of the city of Townsville in North Queensland, Australia. Currently, AIMS has a couple of data loggers situated on the reef that records temperature at two separate depths once every thirty minutes. Scientists from AIMS need to visit the reef periodically to download the data from the loggers.

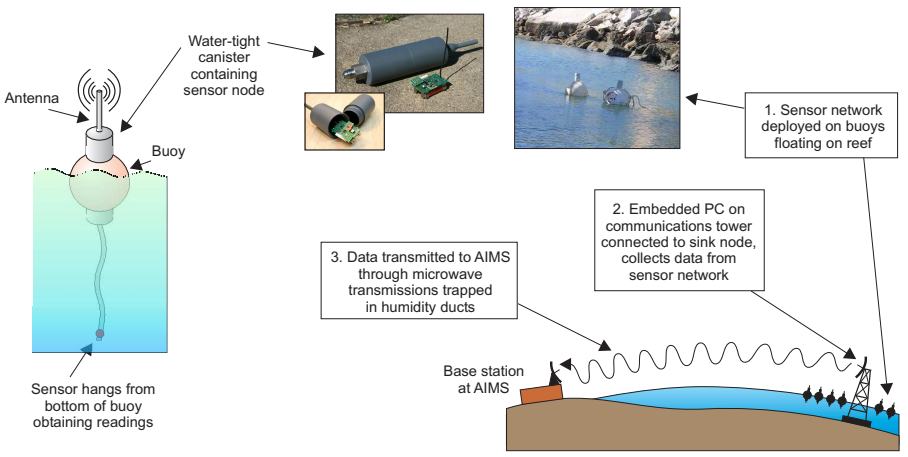

Figure 2.2: Overview of data collection system at Davies Reef

The drawback of the current system is that it only allows single-point measurements. Thus it is impossible to get a true representation of the temperature gradients spanning the entire reef which is around 7km in length. This is because the scale of the fluctuation of environmental parameters in the GBR, ranges from kilometre-wide oceanic mixing to millimetre-scale inter-skeletal currents. Also, the practice of collecting the data once every few weeks makes it impossible to study the trends of various parameters in real-time. Deploying a sensor network would not only allow high resolution monitoring in both the spatial and temporal dimensions but would also enable scientists to improve their understanding of the complex environmental processes by studying data streaming in from the reef in real-time.

The new data collection system that we are deploying at Davies reef can be broken down into three main components as shown in Figure 2.2:

- *Ambient µNodes*: These are the sensor nodes from Ambient Systems [34] that will be placed in water and shock-proof canisters and then placed in buoys around the reef.

- *Embedded PC*: An embedded PC will be placed on a communication tower and will act as the sink node collecting data from all the sensors in the reef.

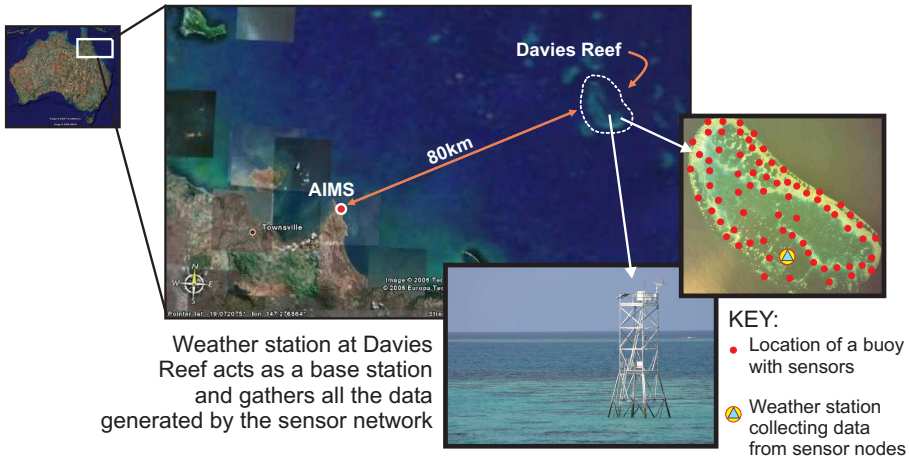

Figure 2.3: Data collected by the sensor network is transmitted to AIMS via a microwave link (Adapted from [30])

- *Microwave link* This will allow data to be transmitted from the Embedded PC to the AIMS base station 80 km away using microwave transmissions trapped inside humidity ducts that form directly above the surface of the sea [117]. This is illustrated in Figure 2.3.

The algorithms presented in this thesis are designed for the first component, i.e. Ambient μNodes.

2.3.2 Space exploration

NASA was one of the first organisations to float the idea of using *sensor webs* [24] to explore the environmental conditions on the surface of other planets or asteroids. While the idea of using wireless sensing devices for space exploration was truly revolutionary, the basic architecture of first generation sensor webs is not very energy efficient. A sensor web is generally made up of a large number of *pods*. A pod is an enclosed casing that contains the sensors and also mechanism for power generation, e.g. a solar panel. The main idea is for every pod to know what is going on in every other pod throughout the sensor web in every measurement cycle. While such a design can be beneficial for tracking the direction of a plume of smoke for example, this architecture is not energy-efficient simply

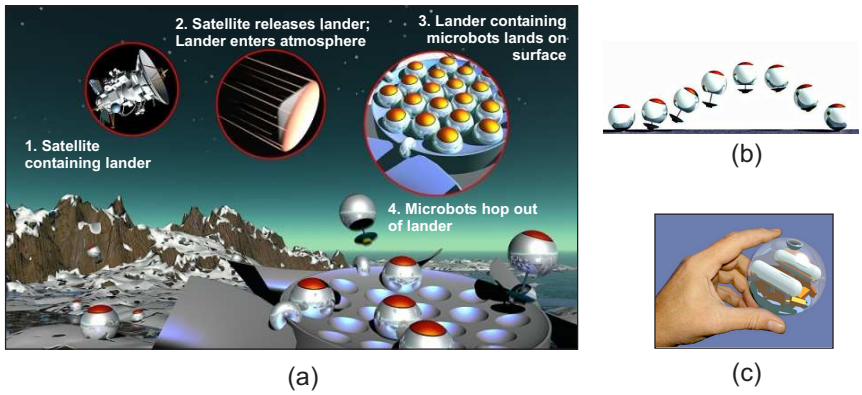

Figure 2.4: (a) Deployment process of microbots, (b) Hopping action, (c) Size of microbot (Adapted from [14].)

because the data gathered by a single pod needs to be propagated to every other pod in the sensor web.

We now describe a space exploration application that makes use of our energy-efficient WSN-based approach described in this thesis instead of the architecture proposed by NASA's sensor webs.

NASA has already sent a number of missions to Mars where rovers have been deployed to monitor a whole range of environmental parameters [12]. While these missions have been successful in returning a lot of data to scientists on Earth, having just a couple of rovers does not allow monitoring to be carried out at a micro-scale. Long-term micro-scale monitoring can be used to study seasonal weather patterns in certain locations on other planets or asteroids and even help in allowing spacecrafts to carry out precision landing operations.

Unlike the GBR application mentioned earlier where sensor nodes are static, in this application, we consider a scenario where mobile *microbots* are spread over a large area on the surface of a planet [14]. As shown in Figure 2.4, a microbot is a small spherical mobile robot that uses hopping and rolling mechanisms to reach scientifically interesting areas in a very rugged terrain. This form of locomotion allows it to explore areas that are inaccessible by conventional rovers that are currently in use on Mars. The device is around 10cm in diameter and is expected to weigh less than 100 grams and will contain a host of sensors that could measure temperature, detect chemicals or even capture images. Also, instead of just deploying a couple of rovers, the idea is to deploy

hundreds or maybe even thousands of microbots on the surface of the planet or asteroid. Such a system is obviously better suited for dangerous space missions than using rovers, as the loss of a few microbots would not hamper the entire project. The microbots may be deployed using an airbag landing system. This is similar to the techniques that have already been used in previous missions to Mars. Once deployed, the microbots would create a multihop network to communicate and transmit all acquired data through this network to the lander. The lander will then send the data to Earth via a satellite circling Mars.

2.3.3 Monitoring a tropical rainforest

Bukit Timah Nature Reserve (BTNR) is the only primary rainforest in Singapore, housing innumerable species of tropical trees and animals, of which a number are in the endangered list [110]. Originally part of the larger Central Water Catchment area, this forest was truncated from the bigger part of the forest in 1985 when the Bukit Timah Expressway (BKE) was constructed to run right through the heart of the forest. As a result BTNR was physically fragmented.

Presently the Nature Reserve has a designated size of some 180 ha of land area although the real forest is only about 75 hectares. The forest area is bounded by newly developed urban land use such as condominiums, open areas, transport infrastructure (highways, roads, railway lines), and closed quarries (Figure 2.5). But parts of the inner forest still maintain its primeval characteristics and it offers the urban population much respite from the stresses of city life. With initiatives from the National Parks Board which manages the forest, it has gone through some major facelift in terms of facilities aimed at attracting visitors and it sees large numbers of visitors, mostly locals from the neighbourhood and organised groups from schools and other organisations. It also provides opportunities for local schools, universities and other research bodies to conduct forest ecology related research. The forest has therefore moved from an unknown forest to one that provides services such as recreation, education, and research opportunities.

However, in land-starved Singapore the proximity to Nature areas is a great attraction for residential home buyers' market. This has created opportunities for private land developers to develop residential properties close to the Nature Reserve, just outside the officially designated boundary of the forest. Several high-rise and low-rise condominiums have been developed within 200 meter periphery of the forest. The boundaries also have other urban infrastructural set-ups such as major roads, a railway line and even a golf course around it.

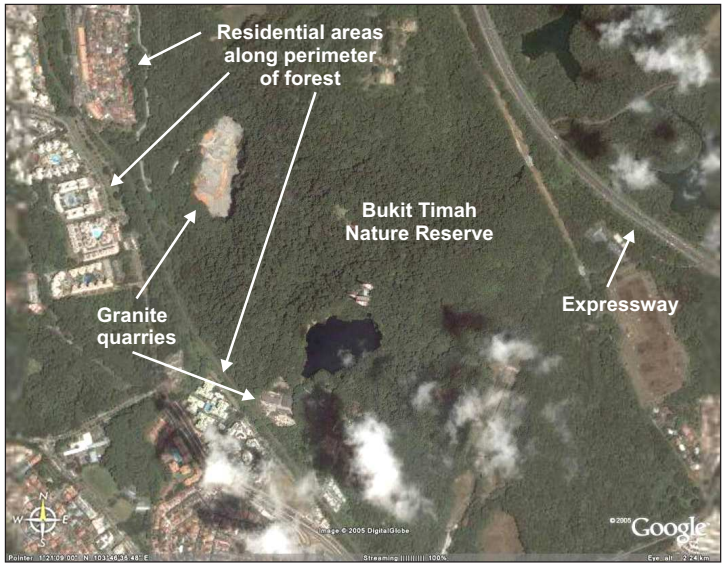

Figure 2.5: Bukit Timah Nature Reserve (Adapted from [30].)

The granite quarry adjoining the forest has been closed and at present does not operate but remains as an open space.

Introduction of such alien non-forested habitats around the forest has introduced some environmental stresses along with the spatially divisive and enveloping fragmentation [116]. Apart from being separated from the bigger body of forested land, the inhospitable surrounding environment alters the physical conditions in the forest fringe and poses a great threat to the survival of the only remnant primary forest that is so much an emblem of tropical Singapore.

Urban built up areas, road development, non-forest open areas around the forest alter the peripheral environmental conditions and make the environmental gradient between the interior and the periphery very steep. Increased exposure to sunlight, higher temperature, wind conditions, and lower relative humidity at the exposed peripheries impact on the soil conditions. Desiccated top soil affects the biological environment and leads to steep gradients in the moisture status of the soil profile at the forest edges. These extreme conditions at the periphery make the interior to exterior environmental gradient very steep. Increasingly such inhospitable conditions encroach into the forest inner areas as more inroads

are made into forested areas. Extensions in such ecologically stressed areas around the forest alter the edge/interior ratio, causing irreparable damage to the forest interior. This makes the core of the forest vulnerable and in turn may affect the quality of the forest itself. The elongated shape of the forest makes it more vulnerable as edges are close to the inner core areas in places where the width of the forested segment is less, increasing the effects of the steep environmental gradient.

As a result of the fragmentation and intrusion of alien open environment the forest at BTNR has become a truncated forest fragment and is in danger of decay. The small area and dismemberment from the main forest body will have long term adverse effect on the biological status of the forest, deteriorating the original forest conditions. Certain segments of the forest may be more affected than others. Determination of higher stress areas are of immediate importance. Truncation has also resulted in reduction of animal migration between the severed segments of the original forest. As tropical rainforests are heavily dependent on animals for survival, these restrictions imposed by fragmentation are bound to affect the nature of the forest.

The deployment of a WSN at the BTNR will allow the various areas of the forest to be monitored at high spatial and temporal resolutions. Placing sensors along various transects throughout the forest will enable researchers to understand the degree of difference between environmental conditions in the core of the forest and in the periphery areas. They will also help identify areas that are under extreme stress and will allow town planners to make more informed decisions when deciding on future infrastructure development that may have a detrimental effect on the forest reserve. Forest rangers will also be able restrict visitors from areas of the forest that are deemed to be under stress. Sensor networks will also be deployed to ascertain animal movement paths between the fragmented BTNR and the central forest for creating ecological corridors between the two separated parts of the forest. Conservation of BTNR through management of the peripheral zones and by re-establishment of animal movement pathways are crucial to the sustenance of the central forested area in Singapore, particularly since this acts as the only water catchment in water-scarce Singapore.

The data collected using the WSN in BTNR will have multiple data consumers. Firstly environmentalists will pose queries to find out the precise impacts on the forest due to peripheral developments. Civil engineers will use the network to understand the nature of slope stability both along the peripheries and within the inner areas of the forest. Apart from the research community, the WSN will also be accessible to students both at the tertiary and high school

levels to help them carry out various projects on sustainable development and provisions will also be made for the public to pose queries through the Internet. Thus while the network will mainly be used for research, it will also be used as an educational tool.

2.4 Essential characteristics of protocols designed for WSNs

Based on the requirements of the applications described in Section 2.3, we now state some of the properties that should exist in protocols designed for such applications.

- *Query processing capability*: The algorithms in these applications are designed for two types of querying scenarios. The first scenario refers to *long-running queries*. Such queries require all nodes to sample their sensors at a specified frequency. This technique can also be referred to as *raw data collection*. Long running queries are used in the GBR and space exploration applications. The second scenario refers to *snap-shot range queries*. These queries require nodes to transmit data at a particular time only if certain specified conditions have been met. The BTNR application makes use of snap-shot range queries.

- *Energy-efficient*: It is obvious that all the applications mentioned earlier require the WSN to be operational as long as possible. This makes it essential for the nodes to operate in an energy-efficient manner. Thus the data management and communication protocols running on a node need to ensure that the energy consumed is kept to a minimum by minimising the number of messages that are transmitted and also by ensuring that the sensors are sampled in an energy-efficient manner. Also, a node should analyse every message passing through it to see whether any appropriate steps can be taken to improve energy-efficiency.

- *Self-configuring*: Sensor network applications generally involve a large number of nodes deployed over a wide area and very often, in harsh environments. Under such circumstances, it would be difficult or maybe even impossible for a user to manually configure each node while starting up the network. In certain cases, like in the space exploration application, a user will not even be present. Thus every node should be able to configure itself automatically.

- *Self-stabilising*: Due to the harsh environmental conditions of the deployment area, a certain proportion of the nodes may fail over time. The self-stabilising property would ensure that the WSN is able to automatically recover following the occurrence of transient faults, e.g. one or more nodes get damaged and cease to operate. In certain cases, a network may be extended in stages by adding more nodes. Such a scenario would also require both the old and new nodes to establish communication with each other and operate seamlessly without any manual configuration. In the space application, some nodes may migrate to different areas at different times. Self-stabilising algorithms would then be required so that the nodes are autonomously able to cope with topology changes.

- *Distributed*: It is essential to use distributed algorithms when dealing with large-scale WSNs. While centralised algorithms may result in optimal solutions as they have a global view of the network, they require a large amount of information to be transmitted to the central node. This is not energy-efficient and also may lead to bottlenecks within the network due to the limited bandwidth of present day sensor nodes. Thus centralised architectures are clearly not scalable when it comes to large networks. Furthermore, distributed mechanisms are more robust than centralised approaches due to the lack of a central point of failure. Thus WSNs should aim to use distributed algorithms as far as possible.

- *Adaptive*: A single protocol on a sensor node cannot be suitable for all possible WSN applications as different applications could have completely opposite requirements. Thus protocols for WSNs are generally application specific. However, this does not imply that the protocols should be completely rigid. Instead, in order to operate energy-efficiently, the protocols should be able to adapt within the boundaries of the application domain. For example, a node operating in a dormant/low power state would have to increase its duty cycle when an interesting event has occurred.

- *Small footprint*: As sensor nodes have highly limited memory and computational resources, decisions should be taken only based on locally available information (e.g from a node's immediate neighbourhood) instead of attempting to act on data gathered from all the nodes in the network.

2.5 The cross-layered approach

In this thesis we propose an approach that tries to maximise the benefits that can be derived by using the information provided by a particular layer, in as many ways as possible. More specifically, we illustrate how information provided by the MAC layer is used for the MAC, topology monitoring, routing, localization, time synchronisation, data aggregation, and even sensor sampling. If we followed the traditional OSI approach, every layer would operate independently and transmit its own set of messages. In the following subsections, we first describe the MAC protocol we have selected and then explain the benefits derived from its usage.

2.5.1 LMAC: A Lightweight Medium Access Control Protocol

LMAC [78] is a TDMA-based lightweight medium access control protocol designed specifically for wireless sensor networks. Instead of contending for the medium like carrier-sensing based MAC protocols [148, 59], time in LMAC is divided into frames, each of which is further divided into a fixed number of time slots (Figure 2.6). Every node chooses its own slot using a distributed algorithm that uses only locally available information. A node is allowed to pick any slot as long as it is not owned by any other node within its two-hop neighborhood. This mechanism effectively helps 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. It also prevents all slots from being used up as LMAC ensures that two nodes that are at least 3 hops away from each other can reuse the same time slot.

A time slot consists of two sections, the Control Message (CM) and the Data Message (DM). The CM, which contains control information and has a fixed length, 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. The CM contains a table which indicates the slots that are occupied by itself and its one-hop neighbors and other control information. Table 2.3 lists the fields that are present in the CM and also mentions the number of bits taken up by each field. Every node maintains a *Neighbor Table* that stores the information about its one-hop neighbors, e.g. ID, occupied slot, number of hops to sink node, etc. Occupied slots are marked with a 1 where as unoccupied ones are marked with a 0. A node joining the network first listens out for the CMs of all its neighbors and then picks one of the slots that is marked as unoccupied by performing an

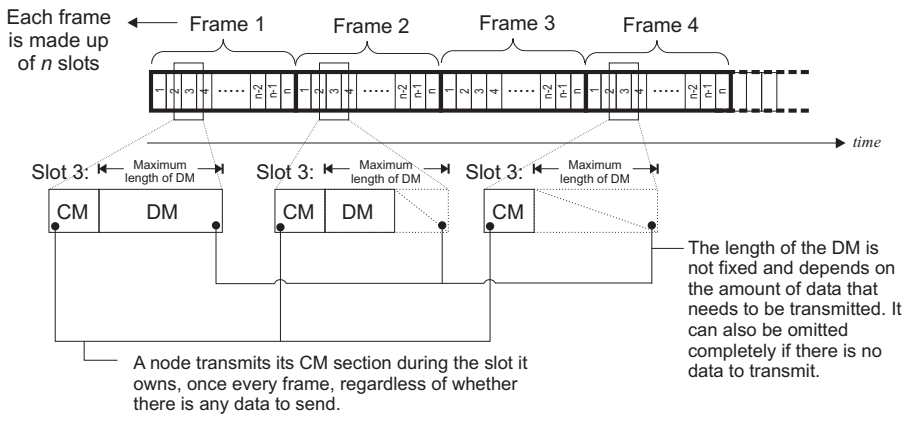

Figure 2.6: Illustration of frames and slots in LMAC

OR-operation. This mechanism is illustrated in Figure 2.7.

The DM contains higher layer protocol messages. The length of the DM can vary depending on the amount of data that a node needs to send. It does however, have a maximum length as shown in Figure 2.6.

2.5.2 Maximising benefits by using cross-layer information

It can be observed from Section 2.5.1 that LMAC provides four useful pieces of information: (i) Neighbourhood information, ***NI***, (ii) A sense of synchronised time - due to the frames and time slots, ***ST***, (iii) Slots assigned to immediate neighbours, ***SA***, (iv) Distance to gateway, ***DG***. Let us now briefly describe how each of these pieces of information is used in the various components of the sensor node architecture. Table 2.4 summarises the points presented below.

- *MAC*: A node running LMAC uses ***NI***, ***ST*** and ***SA*** in order to choose an unoccupied slot.
- *Routing*: A node uses ***NI*** and ***DG*** when choosing a parent node.
- *Scheduling for data aggregation*: Chapter 5 of this thesis illustrates how ***NI*** and ***SA*** are used to assign schedules which decide when a node should perform aggregation.

Description	Size (bits)
Identification	16
Current slot	5
Distance to gateway	8
Occupied slots	32
Collision in slot	5
Destination ID	16
Acknowledgement	32
Total	114

Table 2.3: Contents of the Control Message (CM) - Note that we are assuming that there are 32 time slots in a frame [138]

- *Scheduling for sensor sampling*: Chapter 6 of this thesis illustrates how ***NI*** is used to assign schedules which decide when a node should sample its sensor.
- *Time synchronization*: The periodic broadcast of CMs is used to ensure that time is synchronised among all nodes in the WSN. We refer the reader to [138] for details regarding the synchronisation mechanism as it lies outside the scope of this thesis.
- *Topology monitoring*: A node uses ***NI*** and ***ST*** to detect changes in its neighbourhood.
- *Localisation*: A node can estimate its location by using connectivity information, i.e. ***NI*** [65] or ***DG*** [111].

2.6 Conclusion

In this chapter we have provided the reader with a general overview of what WSNs are. We first started off with the basic components that make up a sensor node platform and then described where WSNs can be used in general. We then described three specific applications that are relevant to this thesis. These applications helped us define the essential characteristics that sensor network protocols should have. We have ensured that these fundamental characteristics exist in the various algorithms we present in the following chapters. Finally, as

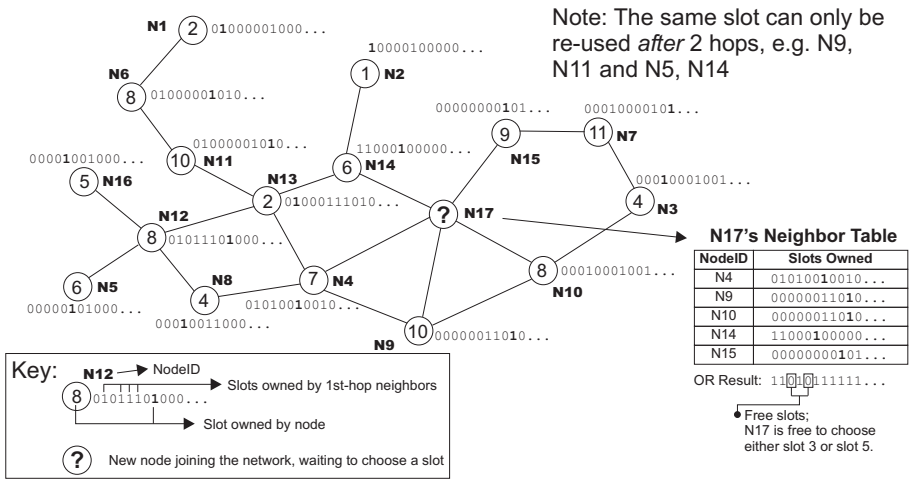

Figure 2.7: Distributed slot allocation in LMAC

we strongly promote the need to follow a cross-layered approach when designing WSN protocols, we first describe the MAC protocol which we have based most of our work on and then highlight the motivation for following this cross-layered strategy.

		Information from LMAC			
		NI	***ST***	***SA***	***DG***
1.	MAC	×	×	×	
2.	Routing	×			×
3.	Scheduling for data aggregation	×		×	
4.	Scheduling for sensor sampling	×			
5.	Time synchronisation		×		
6.	Topology monitoring	×	×		
7.	Localisation	×			×

Table 2.4: How information in the MAC layer is used in various components of the WSN architecture (***NI***: Neighbour information, ***ST***: Synchronised time, ***SA***: Slot assignment, ***DG***: Distance to gateway)

Chapter 3

A Taxonomy of Distributed Query Management Techniques for Wireless Sensor Networks

In the not too distant future, thousands or even millions of sensor nodes may span vast geographical areas enabling various environmental parameters to be monitored with significantly higher spatial and temporal resolutions than what is achievable using existing monitoring technologies. In order to manage the large amount of data that will be generated by these numerous sensor nodes, novel querying methods are needed to extract the required information in an energy-efficient manner. This chapter studies the techniques used to manage the queries in a distributed manner and classifies the current state-of-the-art in this field into four main categories: in-network processing, acquisitional query processing, cross-layer optimisation

and data-centric data/query dissemination[1,2]. This taxonomy not only illustrates how query management techniques have advanced over the recent past but also allows researchers to identify the relevant features when designing sensor networks for different applications.

[1]S. Chatterjea and P. Havinga. A Taxonomy of Distributed Query Management Techniques for Wireless Sensor Networks. In *International Journal of Communication Systems*, 20 (7). pp. 889-908, Wiley, 2006.

[2]S. Dulman, S. Chatterjea, T. Hoffmeijer, P. Havinga, and J. Hurink. Architectures for Wireless Sensor Networks. In *Embedded Systems Handbook, (R. Zurawski ed.)*, CRC Press, August 2005.

3.1 Introduction

The last few years have seen the field of WSNs transform into a multi-faceted research area spanning across a wide range of disciplines within the field of computer science. With the pioneering work focussing on the hardware aspects, the data-centric nature of WSNs and various communication-protocol related issues, sensor network research is gradually branching out into other diverse fields.

Researchers from the database community have added yet another interesting perspective by introducing the idea of viewing a WSN from the point of view of a database management system. The primary motivation for this approach is to help create an abstraction between the end-user and the sensor nodes thus allowing the user to solely concentrate on the data that needs to be collected rather than bothering with the intricacies of mechanisms deciding how to extract data from a network in the most energy-efficient manner.

While there are many concepts from the field of databases that can be beneficial to the area of sensor networks, the unique nature of sensor networks gives rise to a whole new set of issues that makes it inappropriate to apply concepts from databases directly. However, it is possible to adapt existing database solutions to a certain extent to suit the requirements of WSNs.

This chapter introduces the reader to the various distributed query management techniques designed specifically for WSNs proposed thus far and classify them into specific categories. We first highlight the four essential building blocks which constitute a distributed query management framework for WSNs in Figure 3.1. We then go on to describe the state-of-the-art for each category and discuss the salient points of the various query management methods. We conclude the chapter in Section 3.3.

Name of the project/ mechanism/ author	Characteristics							
	Essential conceptual features				Platform			
	In-network processing	Acquisitional query processing	Cross-layer optimisation	Data-centric query/data dissemination	Simu-lation	Mote-class	PDA-class	PC-class
Directed Diffusion, 1999	• Uses filters • Suppression of duplicate messages	×	×	• Publish/ subscribe API based on named data • Gradients used to route data from source to sink		✓	✓	
COUGAR, 2000	• Aggregation at cluster leaders • Packet merging	×	×	×			✓	
TinyDB, 2002	• Evaluation of operators such as MIN, MAX, COUNT, AVERAGE, etc.	• Centralised query optimisation based on collected metadata • Priority given to cheapest sensor when evaluating multiple predicates • Multi-query optimisation: Grouping of multiple identical events	• Communication scheduling primarily for upstream data flow • Nodes able to decide on schedule themselves using hop count • Minimises burden on MAC • Scheme unusable for multiple concurrent queries with different epoch requirements	• Semantic routing • Designed for range queries • Maintenance algorithm may be too costly if measured attribute changes too rapidly	✓	✓		

Continued on next page...

Name of the project/ mechanism/ author	Characteristics							
	Essential conceptual features				Platform			
	In-network processing	Acquisitional query processing	Cross-layer optimisation	Data-centric query/data dissemination	Simu-lation	Mote-class	PDA-class	PC-class
Bonfils *et al.*, 2003	• Decentralised query operator placement	×	×	×	✓			
TiNA, 2003	• Exploits temporal correlations • Readings transmitted only when user-specified threshold is exceeded	×	×	×	✓			
WaveScheduling, 2004	×	×	• Nodes scheduled to minimise MAC collisions • Saves energy, but increases latency	×	✓			
Coman *et al.*, 2005	×	×	×	• Queries forwarded based on spatial and temporal ranges	✓			
REED *et al.*, 2005	• Joins between static, external tables and sensor nodes	×	×	×	✓	✓		
BBQ, 2004	×	• Minimise sampling by predicting sensor readings using centralised statistical models • Identify temporal and spatial correlations	×	×	✓			

Continued on next page...

Name of the project/ mechanism/ author	Characteristics							
	Essential conceptual features				Platform			
	In-network processing	Acquisitional query processing	Cross-layer optimisation	Data-centric query/data dissemination	Simu-lation	Mote-class	PDA-class	PC-class
Ken, 2006	• Nodes keep local model synchronised with central model • Able to detect outliers	×	×	×	✓			
TD-DES, 2003	×	×	• Event scheduler dynamically allocates and multiplexes upstream and downstream time slots for each event type • Root works out schedule	×	✓			
DTA, 2004	×	×	• DTA defines rules controlling order of data transmission • Root works out schedule	×	✓			✓
AI-LMAC, 2004	×	×	• Adapts operation of MAC based on requirements of application • Distributed bandwidth allocation based on expected traffic for a particular query • MAC operates with low duty-cycle in inactive areas of network	• Directs queries only to relevant parts of the network • Queries directed based on attribute name and value	✓			

Continued on next page...

Name of the project/ mechanism/ author	Characteristics							
	Essential conceptual features				Platform			
	In-network processing	Acquisitional query processing	Cross-layer optimisation	Data-centric query/data dissemination	Simu-lation	Mote-class	PDA-class	PC-class
GHT, 2002	×	×	×	• Hashes attribute name into geographic coordinates • Nodes must be location aware • Uses perimeter refresh protocol to improve robustness • Uses structured replication to eliminate bottlenecks during occurrence of multiple identical events	✓			✓
DIFS, 2003	×	×	×	• Supports range queries • Hashes attribute name and value into geographic coordinates • Nodes must be location aware • Does not deal adequately with node failures and packet losses	✓			

Continued on next page...

Name of the project/ mechanism/ author	Characteristics							
	Essential conceptual features				Platform			
	In-network processing	Acquisitional query processing	Cross-layer optimisation	Data-centric query/data dissemination	Simu-lation	Mote-class	PDA-class	PC-class
DIM, 2003	×	×	×	• Supports range queries • Hashes event to a zone • Nodes must be location aware • Re-assigns zones when nodes enter or leave • ACK scheme improves robustness			✓	✓

Table 3.1: A summary of distributed data management techniques for WSNs

3.2 Essential conceptual building blocks

While there are a host of features that are necessary to form a full-fledged distributed data management system for WSNs, we highlight four essential conceptual building blocks in Figure 3.1 that have been widely mentioned in the existing literature. Below, we provide a very brief overview of each building block and describe their specific roles. This is followed by a discussion detailing how the various components are related to each other thus enabling the reader to visualise how the various components fit into the "bigger picture". Table 3.1 provides a summarised comparison of the primary features of some of the key research papers published in the field of distributed query management for WSNs.

- *In-network processing*: This involves moving various types of computation that are traditionally done on the server-side to within the sensor network itself. It is generally used for filtering and processing of messages flowing within the network thus preventing the transmission of unnecessary information.

- *Acquisitional query processing*: Energy consumption in sensor nodes depends on two main factors: Operation of the transceiver and operation of the sensors. Acquisitional query processing helps to minimise energy consumption by targeting the sensors, i.e. sampling of the various attached sensors is carried out in an energy-efficient manner. For example, a user may be presented with sensor readings generated using certain statistical methods rather than actually sampling sensors.

- *Cross-layer optimisation*: Unlike conventional computer networks which can generally be used to perform a wide variety of tasks, WSNs are usually designed for a particular application. This makes it possible to design the various components of the WSN architecture, e.g. the routing and MAC protocols specifically for the application in mind. This could mean that the MAC and routing protocols may be able to adapt to the changing requirements of the application. This is fundamentally different from the conventional OSI model used for typical networks such as the Internet, where the lower-layer protocols operate completely independently from the higher layer protocols.

- *Data-centric data/query dissemination*: Unlike conventional routing protocols which do not actually bother about the content of the data message

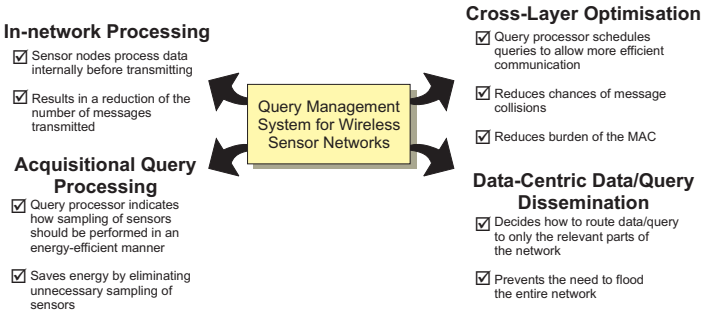

Figure 3.1: Essential building blocks of a distributed query management system for wireless sensor networks

being transmitted, the path taken by a message being routed by a data-centric data/query dissemination protocol for a WSN is dependent on the contents of the message. This allows messages to be routed more efficiently.

While Figure 3.1 illustrates the pertinent features of every building block, it does not illustrate the relationship between them. We present this relationship in Figure 3.2. Probably the most noticeable feature is that we have placed Acquisitional Query Processing, Cross-layer Optimisation and Data-Centric Data/Query Dissemination all within the class of In-Network Processing. The reason for this can be traced back to the way data is usually collected in conventional databases using the warehousing approach. Under this model, data is initially collected from various sources (e.g. sensors with wireless transmitters) and stored at a central location. The data is then processed centrally to extract the required information. This model is highly unsuitable for WSNs as it involves excessive transmission overhead and also prevents users from accessing real-time, streaming data. The only viable alternative is to migrate from off-line processing to processing the data within the network as close to the data source as possible. The practice of processing data at the sensor nodes themselves is known as in-network processing and can result in a significant reduction in the number of messages transmitted. The ability to perform in-network processing is a cornerstone of WSNs.

In order to distinguish between sensors attached with wireless transmitters and wireless sensor nodes, which are actually capable of performing simple computations within them, and also because in-network processing is a very generic

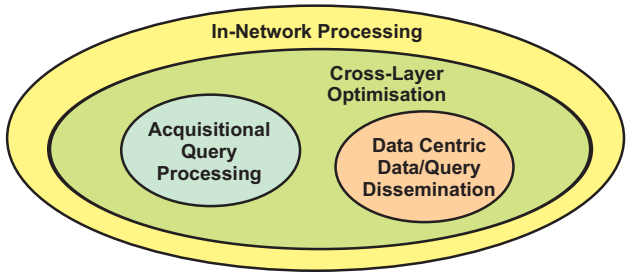

Figure 3.2: Relationship between the building blocks

term (i.e. it could essentially refer to any sort of processing that takes place within a node) we have placed Acquisitional Query Processing, Cross-layer Optimisation and Data-Centric Data/Query Dissemination all within the class of In-Network Processing. All these three sub-classes require a sensor node to analyse the data it has at hand and make certain decisions based on the outcome of the analysis instead of transmitting all the data to a central server for offline processing or centralised decision-making. Such processing is not possible in normal sensors that only have attached wireless transmitters. The kind of processing that can take place within a wireless sensor node varies greatly, which is why we have these three other sub-categories.

Cross-layer optimization generally refers to schemes where the application influences the operation of the routing or MAC layers. In the Cross-Layer Optimization category, we cover literature which studies the relationship between the MAC and the application layers. Data-Centric Data/Query Dissemination is classified as a sub-class of Cross-Layer Optimization as it involves creating links between the application and routing layers. Acquisitional Query Processing falls under Cross-Layer Optimization as it deals with schemes where sensor sampling can be dependent on information gathered from the injected query (i.e. application layer) or even the underlying MAC.

In order to simplify matters and also because the features present in a particular class are not present in its super class, we analyse each building block separately in the following subsections. However, the reader should keep in mind the existing relationships indicated in Figure 3.2. The following subsections describe each category in greater detail and illustrate the different methods that have been employed to incorporate the various features.

3.2.1 In-network processing

While the precise manner in which in-network processing is carried out on sensor nodes may differ between various publications, the fundamental objective is still the same - to save energy by reducing message transmissions. Directed Diffusion [82] performs in-network processing using filters. When a node receives a message that matches its filter, the message is first handed over to the application module within the node for processing instead of forwarding it to the next node. For example, the application might carry out a suppression of duplicate messages indicating the detection of the occurrence of an event so as to prevent a sudden burst of identical messages when a bunch of nodes detect the same event. The main drawback of Directed Diffusion, however, is that it has a non-declarative interface and thus does not rank very highly in terms of usability.

Cougar [147], which was one of the first projects to view a sensor network as a distributed database with a declarative interface, uses a clustered approach to in-network processing. A network may consist of several clusters each of which is made up of a single leader node and a group of child sensor nodes belonging to the leader. Child nodes periodically send their readings to the leader node which then aggregates the received readings. The leader then forwards the computed result toward the root of the network. Computation at the leader only takes place once all the child nodes have responded. Additionally, since sending multiple small packets is more expensive than sending one larger packet (due to the packet header payload and the cost of reserving the medium) Cougar performs packet merging by combining several packets into one. This method is particularly beneficial when servicing queries that generate holistic aggregates where intermediate nodes cannot perform any partial aggregation and all data must be brought together to be aggregated by the node evaluating the query, e.g. the Median operator or even the collection of raw sensor readings. It is important to note, however, that while Cougar claims to be designed for WSNs, it was deployed on PDA-class devices that had significantly larger processing power and could even run Windows CE and Linux [95]. Their design does not consider the impoverished power and computational constraints of conventional sensor nodes, e.g. XML, which is known for its verbosity, is used to encode messages. Apart from packet merging, Cougar lacks any other features that make it energy-efficient. It also fails to address the issues of reliability and self-organisation and relies instead on the underlying 802.11 MAC protocol.

TinyDB [95] supports a number of aggregation operations (e.g. MIN, MAX, SUM, COUNT, AVERAGE, etc.) over certain user-specified sample intervals.

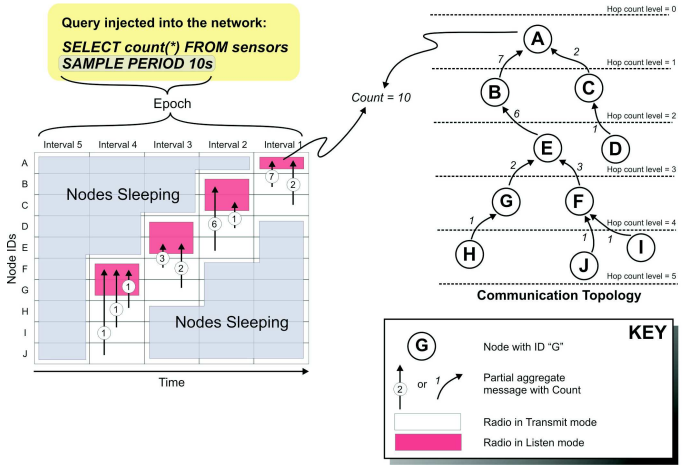

Figure 3.3: An aggregation operation using interval-based communication scheduling (Adapted from [94].)

As sensor readings flow up the communication tree, they are aggregated by intermediate nodes that are able to meet the requirements of the query (Figure 3.3). Without aggregation, every node in the network needs to transmit not only its own reading but also those of all its children. This causes a bottleneck close to the root node and also results in unequal energy consumption, i.e. the closer a node is to the root node, the larger the number of messages it needs to transmit which naturally results in higher energy consumption. Thus nodes closer to the root node die earlier. Losing nodes close to the root node can have disastrous consequences on the network due to network partitioning. Using in-network aggregation, however, every intermediate node aggregates its own reading with that of its children and eventually transmits only one combined result. This also naturally implies that the size of the message remains constant as it traverses from the source nodes to the root. TinyDB also illustrates how aggregation can be extended to perform more complex tasks such as vehicle tracking and isobar mapping [75]. TinyDB depends on MAC-level acknowledgements and runs a tree-maintenance algorithm continuously in order to ensure that communication between nodes can continue seamlessly at all times.

The Temporal Coherency-Aware In-Network Aggregation (TiNA) [128] scheme

improves upon TinyDB as it allows users to specify a threshold which in turn is used to reduce the amount of information that is transmitted by individual nodes. Instead of transmitting every reading at fixed intervals (like in TinyDB), TiNA exploits temporal correlations of readings to suppress certain messages, i.e. if the newly acquired reading differs from the previous reading by an amount that is less than the user-specified threshold, the reading is not transmitted. In order to ensure reliability, nodes in TiNA send periodic heartbeat messages to their parent nodes so that the parent node knows that the child node is alive during periods when the acquired readings are fairly constant and fall within the user-specified threshold.

Bonfils and Bonnet [38] describe another form of in-network processing that allows a node to run a completely localised algorithm (based on information from one-hop neighbours) that ensures that in-network processing is carried out in an efficient manner. This is done by placing aggregation operators such that the amount of data that needs to be transmitted is minimised. Once the operators have been placed on an arbitrary node, the localised algorithm ensures that the operator progressively reaches its local optimal placement by greedily moving to the neighbour with the lowest estimated cost. This method is particularly suited for handling holistic aggregates. It is also useful for carrying out join operations between tables stored on pairs of sensors in a network. While the work allows nodes to take decisions in a decentralised manner, there is no mention of any fault tolerant operator placement algorithms.

Rather than focussing on joins between pairs of sensors, REED (Robust, Efficient Filtering and Event Detection in Sensor Networks) [31] concentrates on joins between static external tables (i.e. tables stored outside the network) storing filter conditions and sensor nodes within the network. External tables are typically injected into the network along with the query. If a receiving node is able to store the entire table in its limited memory, the node performs a join operation and returns the result to the root node. In the event that the external table is too large to be accommodated within a single node, REED partitions the table horizontally into smaller fragments. The fragments are then disseminated to a group of nodes (which are within close proximity of one another) such that each node in the group stores a single fragment. The join operation is performed within the group and the result is subsequently returned to the root. REED can also transmit fragments of the table into the network, forcing nodes which do not have entries in the table to be joined externally. Timeouts and periodic advertisements are used to improve the robustness of the system.

While the majority of the literature in this category focuses on utilising different strategies to ensure energy-efficient operation, none of the authors inves-

tigate the performance of unreliable message transmissions. As current sensor nodes are known to be unreliable and the performance of the various in-network processing schemes can change dramatically with varying degrees of reliability, this remains as a topic which is open to further research.

3.2.2 Acquisitional query processing

In conventional database systems, when posing a query to the system, a user need not bother about how the data should be extracted in the most efficient way. The same concept holds true in distributed database systems where the requested data might be stored in small fragments spanning the entire network. In this case, the user does not require knowledge of where the data resides. In other words, the techniques used to locate data or the methods followed to extract data in an efficient manner, are processes that are completely transparent to the user. The user simply injects the query into the network and waits for the result to appear. It is the responsibility of the query processing system to handle the above-mentioned tasks and act as an interface between the user and the data sources.

One of the tasks of a query processor for WSNs is to generate an optimised query execution plan that outlines an energy-efficient strategy to execute a query. While conventional database query optimisation techniques calculate the cost of executing a query based on a number of parameters such as CPU instructions, I/O operations, messages transmitted, etc. the model necessary for WSNs is slightly different due to its unique characteristics described earlier. The tight power constraints of WSNs have driven researchers to find novel ways to minimise the two main sources of power consumption on a sensor node - the operation of the radio transceiver and the sampling of the sensors to obtain readings. The idea of managing sensor sampling as one of the tasks of the query processor was first introduced in TinyDB [96] and was termed as acquisitional query processing.

TinyDB carries out query optimisation at the root node. Metadata such as the energy and time required to sample a particular sensor, information about the costs of processing and delivering data, etc. are periodically copied from the nodes to the root for use by the query optimiser. The optimiser also has the task of initialising, merging and updating the final value of partial aggregate records as they propagate through the network towards the root.

Before a query is injected into the network through the root node, the query optimiser can optimise the query in various ways using the collected metadata. For example, since the difference in power consumption of sampling various

sensors on a single node can differ by several orders of magnitude, the order in which sensors are sampled when evaluating a query can have a substantial impact on network lifetime.

Consider the scenario where a user requires readings from a light sensor when the temperature and humidity sensors reach thresholds t and h respectively. If the node uses a humidity sensor that requires 100 times more energy to sample than the temperature sensor, in most cases, it would be a lot more energy efficient to sample the temperature sensor first to check if the threshold has been met and proceed with the sampling of the humidity sensor only if the result of the temperature threshold check is found to be True. In fact, in such a scenario, sampling the humidity sensor first could be up to an order of magnitude more expensive.

TinyDB also performs multi-query optimisation on event-based queries in order to reduce costs due to sensor sampling and transmission. Consider a query Q that requests temperature readings only when a certain event E occurs. The occurrence of a string of event Es within a short time interval would trigger multiple instances of the query to run simultaneously. This results in high energy consumption as every instance of a query performs its own sensor sampling and transmission of results. To alleviate this problem TinyDB optimises such queries by rewriting them so that all occurrences of event E of the last k seconds are buffered. When a sensor reading is obtained, it is compared against the buffered events and the temperature readings are returned. Thus no matter how frequently events of type E are triggered, only one query is required to run.

Deshpande et al. [60] extend the initial acquisitional query processing concepts introduced in TinyDB by utilising certain statistical modelling techniques. The Barbie-Q (BBQ) query system creates models that provide answers that are more meaningful and also help to extend the lifetime of the network by returning approximated results coupled with probabilistic confidences. There are 3 major steps involved in query processing in the BBQ architecture:

- Building the model: This involves the collection of raw data from every node. This historical data is used to build a model based on time-varying multivariate Gaussians and allows one to observe the correlations and statistical relationships between sensor readings on various nodes.

- Generating an observation plan: Users inject queries that are similar to SQL except for the fact that they are allowed to include error tolerances and target confidence bounds that specify the degree of uncertainty a user is willing to accept. Using the probabilistic model mentioned earlier,

coupled with the accuracy specifications of the injected query, the system takes the liberty of determining the most efficient way of servicing the query and generates the corresponding observation plan. Depending on the specified error tolerance, the observation plan may only poll a small proportion of all the sensors that are actually capable of servicing the query. It may even retransform a query by querying a different sensor from the one specified in the original injected query, if it is more energy-efficient and both sensor readings are found to be correlated.

- Updating the model: BBQ relies on Markovian models to keep the model updated regarding changing environmental parameters and to ascertain any temporal correlations. This helps to generate a probability density function (pdf) of all parameters at time t+1 given the pdf at time t.

  Thus unlike TinyDB and Cougar, BBQ does not interrogate all sensors every time a query is injected into the network. Instead, sensors are used to acquire data only when the model itself is not sufficiently rich to answer the query with acceptable confidence. As time passes, the model may update its approximations of sensor readings to reflect expected temporal changes in the data.

  As BBQ builds a statistical model centrally, it is unable to detect sudden changes that may occur within the network, e.g. spikes in temperature readings or changes in network topology. Ken [51], which is similar in certain respects to BBQ, tackles the problem of detecting outliers, by storing models within the sensor nodes instead of storing them centrally. Every node also transmits its model to the central server. When a node acquires a reading, it checks the acquired reading against the locally stored model. If the acquired reading falls outside the reading predicted by the model by a margin that is larger than the user-specified threshold, the node re-computes a new model and transmits this new model together with the newly acquired reading. Ken too, however, is unable to adapt to changes in the network topology. It should be noted at this juncture that Ken, however, does not actually fall in the class of Acquisitional Query Processing. Instead it should fall in the super-class, In-Network Processing, as Ken does not actually deal with sensor sampling. Instead, it aims to minimise communication by making use of dynamic probabilistic models. We have, however, mentioned it here as its operation is very closely linked to BBQ.

3.2.3 Cross-layer optimisation

Traditional system architectures are commonly known to adhere to the layered protocol stack design where every layer operates in a completely independent manner. The Internet browser on a PC, for example, does not require any knowledge about the kind of network connectivity available. It works regardless of whether the user is connected via Ethernet or 802.11b. Such a layered approach, however, is not optimal from the energy efficiency point of view especially when considering WSNs. This is because unlike the network being used in an office LAN, WSNs are typically application-specific networks. Thus it only makes sense to try and make the various components of the WSN architecture more application-aware by performing certain cross-layer optimisations thereby helping to improve network lifetime. Note, however, that while the term "cross-layer optimisation" in the field of sensor networks might refer to the optimisation between the application and routing layers for instance, in this case we refer to the more radical approach where optimisation is performed between the application and MAC layer.

As usage of the transceiver is an energy consuming task, it is imperative that maximum benefit is derived during the time it is operational. Thus rather than encountering energy-wasting collisions during data transmission or actively waiting for messages that do not arrive, current cross-layer optimisation techniques use a variety of methods to try and schedule tasks in an energy-efficient manner. We now review a number of these techniques.

TinyDB [96] uses an interval-based communication scheduling protocol to collect data where parent and child nodes receive and send data (respectively) in the tree-based communication protocol. The cross-layer optimisation in TinyDB involves (i) reducing the burden on the MAC using specifications from the injected query and (ii) routing data from the source nodes to the root. Each node is assumed to produce exactly one result per epoch (time between consecutive samples), which must be forwarded all the way to the base station. As shown in Figure 3.3, every epoch is divided into a number of fixed-length intervals which is dependent on the depth of the tree. The intervals are numbered in reverse order such that interval 1 is the last interval in the epoch. Every node in the network is assigned to a specific interval which correlates to its depth in the routing tree. Thus for instance if a particular node is two hops away from the root node, it is assigned the second interval. During its own interval, a node performs the necessary computation, transmits its result and goes back to sleep. In the interval preceding its own, a node sets its radio to "listen" mode collecting results from its child nodes. Thus data flows up the tree in a staggered manner

eventually reaching the root node during interval 1.

While TinyDB's slotted scheduling protocol does help conserve energy by keeping nodes asleep a significant proportion of time, it is primarily designed for servicing a single query posed to the entire network. The scheme is unusable if there are multiple concurrent queries with different epoch requirements.

WaveScheduling [133] is a scheme that helps minimise collisions at the MAC layer by carefully scheduling nodes. While the scheme results in energy savings, the drawback is that latency is increased. The solution is however, a bit "rigid" as the nodes need to be arranged in a grid in order to operate properly. This may not be feasible in all applications.

Cetintemel et al. [44] describe Topology-Divided Dynamic Event Scheduling (TD-DES) which is an event-based communication model for WSNs that allows a node to switch its radio to a low-power standby mode when it is not interested in certain queries/events that have been disseminated into the network. However, rather than being a MAC layer itself, TD-DES is intended as an application overlay to a CSMA/CA wireless MAC layer. TD-DES sets up a wireless multi-hop network tree where the root node generates a data dissemination schedule describing when nodes should switch on their transceivers to capture certain events. This schedule is disseminated throughout the entire network so that nodes can follow it. The schedule is divided into fixed-sized time slots. Each slot is further divided into three sections, a control event receive slot, a control event send slot and an event data slot.

Every node in the network has a specific subscription profile that describes the list of measured parameters or events that the node is interested in. The node may either be a subscriber, i.e. it is interested in a particular event or a generator of certain events. The control event send slot holds scheduling information of events and is transmitted to neighbouring nodes. A node listens out for scheduling information from a neighbouring node during the control event receive slot and the event data slot is used to transfer data regarding the actual event. A node listens to the relevant portion of the event data slot in receive mode only if it discovers an event that is of interest to itself or one of the nodes in its subtree listed in the scheduling information received from a neighbouring node in the control event receive slot.

TD-DES differs from TinyDB's scheduling scheme in the sense that TD-DES addresses both upstream and downstream data dissemination while TinyDB focuses solely on upstream aggregation. Also, while TinyDB assumes that every node in the network takes part in servicing the injected query, in TD-DES, nodes not subscribing to an event do not own any scheduled slot.

Zadorozhny et al. [150] describe a framework that helps to extend the synergy

between the MAC layer and query optimisation. This is achieved with the help of a Data Transmission Algebra (DTA) that provides the semantics for defining rules that control the order in which nodes transmit data. For example, if there is a node that needs to perform task A and another node that needs to perform task B and both tasks require the use of the RF medium, the DTA is used to work out a schedule such that both tasks are performed with minimal chance of collisions from occurring. The DTA includes three basic operations that combine the two transmissions schedules of tasks A and B as follows:

- Either Task A is scheduled to perform before Task B or Task B before Task A
- Task A MUST be performed before Task B
- Task A and Task B can be performed simultaneously (e.g. when nodes involved are not within transmission range of one another)

The primary aim of this framework is for the root node to work out schedules for all nodes in the network using the DTA scheduler. It is then up to the nodes to decide how to behave within a set of constraint intervals specified by the schedule. As the number of possible schedules grows exponentially with the number of sensor nodes, heuristic-based pruning methods are used to eliminate suboptimal alternatives.

The above-mentioned scheduling techniques clearly indicate that current cross-layer optimisation schemes just stop short of actually altering the operation of the MAC protocol itself. Instead the schemes simply employ different strategies to turn the MAC on and off at different times therefore decreasing the burden placed on the MAC. If collisions still do occur, it is the responsibility of the MAC to recover from them.

Chatterjea et al. [50], however, describe an Adaptive, Information-centric and Lightweight MAC protocol for wireless sensor networks (AI-LMAC) that adapts its operation based on the requirements of the application. The amount of bandwidth that is allocated to a node is proportional to the amount of data that is expected to flow through it in response to the query it is servicing. Bandwidth allocation is done in a distributed manner and is not static but changes depending on the injected query. Information about the expected data traffic through a node is obtained using a completely localised data management framework that helps capture information about traffic patterns in the network. A major advantage of this approach is that the MAC protocol reduces its duty cycle on nodes that are not taking part in servicing the query, thus improving

energy-efficiency and limiting communication activity only to areas of the network where it is actually required. The data management framework is also used for efficient query dissemination (i.e. directing queries to only the relevant parts of the network) and query optimisation. Thus cross-layer optimisation in AI-LMAC addresses the entire spectrum of data management issues, i.e. operation of the MAC, routing, and query optimisation.

3.2.4 Data-centric data/query dissemination

Deciding how to route or disseminate data within a communication network is typically performed using IP-style communication where nodes are identified by their end-points which can be directly addressed using an address that is unique to the entire network. Such addressing schemes are used in the Internet or office LAN. Due to their application-specific nature however, WSNs tend to use a data-centric addressing scheme where "names" are used to create an abstraction of node network addresses. For example, instead of requesting the reading of a node with a particular ID, a typical query usually requests for an attribute of a phenomenon instead. Data dissemination schemes generally address three main issues:

- How queries are routed only to the relevant nodes from the node injecting the query into the network (Flooding a query to all nodes is not energy-efficient)
- How results are routed back to the querying node
- Robustness: how the scheme copes with dynamic network topologies, i.e. node failures and node mobility

We review a number of data dissemination schemes which use distributed techniques to ensure that queries only propagate towards sensors capable of serving the injected queries.

Directed Diffusion [82] is one of the pioneering data-centric data dissemination paradigms developed specifically for WSNs. It is based on a publish/subscribe API where the sink injects an interest (e.g. interest for a particular type of named data, such as the "Detection of a bird") into the network. Every receiving node keeps a copy of the interest in its cache. The entry in the cache also stores a gradient that indicates the identity of the neighbouring node that originally sent the interest. Gradients are formed gradually at every node

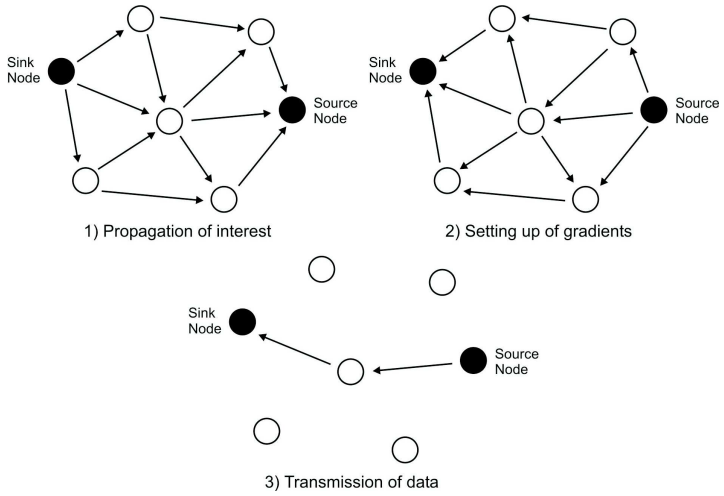

Figure 3.4: Setting up of gradients in Directed Diffusion

as the interest propagates from one node to another eventually flooding the entire network (Figure 3.4). When a source node is able to service the interest, it sends data back to the sink along the path of the gradients set up initially by the interest. In the event data starts flowing toward the sink along multiple gradient paths, the source node reinforces one or a subset of these paths. Reliable paths are usually reinforced while unreliable ones are removed by expiration due to lack of reinforcements or explicit negative reinforcements. Such gradients allow the local repair of failed or degraded paths and do not require the re-flooding of the interest. However, it is necessary to perform flooding when a new interest is injected into the network.

While Directed Diffusion performs routing based on named data, TinyDB performs routing using a Semantic Routing Tree (SRT) which is based on the actual values of sensor readings. It is useful for servicing range queries. An SRT is an index built over some constant attribute A and is stored locally at every node in the network. The index at a node consists of a one-dimensional interval representing the range of A values being generated not just by the node itself but also by all its descendants. When a node encounters a query, it only forwards it to its immediate children which are reported to be transmitting values matching the required range specified in the query. The readings may have been generated either by any of the immediate children or by any of the nodes within the sub-

trees rooted at the immediate children. Additionally, a node executes a query by itself if it can be serviced locally and subsequently transmits the result to its parent. The result eventually propagates up the tree towards the root. If the query cannot be serviced by the node or any of its children, it is dropped. The entry of a child node expires from the SRT of a parent node if the parent node does not receive any updates from the child within a predefined timeout period. The parent then updates its interval information by querying its children and also informs nodes higher up the hierarchy if any changes are detected. While the SRT maintenance algorithm is capable of reflecting changes in the network dynamics (e.g. death of a node), the cost of updating ranges could be prohibitive if the value of the measured attribute changes too rapidly.

Coman et al. [54] present a framework for processing queries that specify the spatial area and the temporal range the answers must belong to. The framework has two phases. In the first phase, a path is searched from the query originator to a sensor node located within the query's spatial window. Next, the located sensor assumes the role of query coordinator and gathers results from all the relevant sensors from within the spatial window and transmits the results back to the query originator. Note that it is assumed that all the nodes are location aware.

Unlike Directed Diffusion and TinyDB's SRT, Ratnasamy et al. [126] deal not just with the data-centric routing aspects but integrate it with the issue of storage within the network. They propose using Data-Centric Storage (DCS) where all events are named. A vital assumption is that all nodes are location aware. There are two main operations within DCS - storing and retrieving data. When a node detects a particular event, it stores the data by name in a node within the network. The Geographic Hash Table (GHT) scheme performs a hashing on the name of the data into geographic coordinates thus deciding in which part of the network data should be stored. GHT is also used in a similar manner when retrieving data. In order to address the issue of robustness, GHT uses the perimeter refresh protocol. This protocol replicates the event data at nodes around the location to which the hashing was originally made thus ensuring that queries can still be serviced even if certain nodes fail. The occurrence of multiple identical events (i.e. events with the same name) would all hash to the same location thus creating a bottleneck in the network for both store and retrieve operations. In order to alleviate this problem, GHT employs structured replication such that events hashing to the same node are mirrored in different parts of the network.

GHT is effective in saving unnecessary transmissions by preventing the need to flood the entire network with queries by performing hashing on an attribute.

However, DIFS [72] achieves even greater savings by also including support for range queries, i.e. queries where only events with attributes in a specific range are required. This is because events defined by attributes with values that fall within a specified range are by definition less common. For example, while there may be many humidity sensors in a network, there may only be a small fraction which have readings higher than a certain threshold. DIFS constructs a multi-rooted hierarchical index where non-root nodes can have multiple parents. Thus if a child node has p number of parents and maintains a range of values r, each index of a parent node maintains an equal proportion of r, i.e. r/p. However, the narrower the value range covered by a node, the wider the spatial extent an index node knows about. In other words, higher-level nodes cover smaller value ranges detected within large geographic regions while lower level nodes cover a wider range of values from within a smaller geographic region. DIFS also reduces the possibility of bottlenecks occurring close to the root node as queries can be injected into any node in the tree. Unlike GHT which can hash to any part of the network, the DIFS hash function restricts its output to the area that a node in the hierarchy has to cover. The function outputs a location upon receiving the following inputs: Event name, Event value and Event location. DIFS however, does not deal adequately with the issue of node failures and packet losses.

The distributed index for multi-dimensional data (DIM) [70] goes a step further by building on top of the above-mentioned methodologies by including support for queries that specify multiple range conditions. For example a query might require all events to be reported that fall within a particular temperature range and a particular humidity range. Such queries can be particularly useful for correlating multiple events and subsequently triggering certain actions. DIM runs a distributed algorithm on every node that eventually divides the sensor field such that there is a single node in each zone. Next, using the values of multiple attributes, the hashing function hashes the event to a zone. The event is subsequently routed to the node that owns the zone and is stored there. DIMs include the following features to address the issue of robustness:

- use of a mechanism to help re-assign zones when nodes join or leave the network
- use of a distributed algorithm to minimise the chance of data loss by carrying out replication of data
- an ACK scheme to improve resilience to packet loss

Query and data dissemination schemes that prevent the need to flood the entire network have progressed markedly in the recent past. From initially just considering event types or ranges of actual sensor readings, they currently support multiple range queries and use various hashing functions to direct queries to the appropriate sections of the network using the attribute types and values as inputs. Furthermore, they incorporate in-network storage as an integral part of the query dissemination mechanism. However, these newer schemes assume that all nodes are location aware thus increasing the complexity of the system.

3.3 Conclusion

As time progresses, WSN deployments are gradually going to grow larger and certain deployments may even be enlarged in stages. This makes it increasingly necessary to improve support for heterogeneous networks, multiple roots and optimisation of multiple simultaneous queries that may overlap partially over sensor types, readings, and spatial and temporal parameters. Cross-layer optimisations from the application layer also need to dig in deeper into the network layers and attempt to eventually influence the operation of the MAC. Query optimisations need to be pushed into the network to prevent large amounts of metadata being sent back to the root. An interesting observation from the summary presented in Table 3.1 shows that only a handful of the projects have actually been implemented on sensor node platforms. This clearly reflects that while certain projects have begun making inroads into the various essential conceptual features mentioned in this chapter, current distributed data management techniques only touch the tip of the iceberg.

Chapter 4

Using Sensor Data for Routing and MAC

This chapter describes two mechanisms using cross-layered optimization strategies where sensor data is used to influence the operation of the routing and MAC layers. We first describe a directed query dissemination scheme, DirQ that instead of flooding, routes queries to the appropriate source nodes based on both static and dynamic-valued attributes such as sensor types and sensor values[1]. Nodes running DirQ are able to adapt autonomously to changes in network topology due to certain cross-layer features that allow it to exchange information with the underlying MAC protocol. DirQ allows nodes to autonomously control the rate of sending update messages in order to keep the routing information updated. Our results show that DirQ spends between 45% and 55% of the cost of tree-based flooding. The second part of the chapter introduces an Adaptive, Information-centric and Lightweight MAC (AI-LMAC) protocol that adapts its operation depending on the amount of data traffic that is expected to flow through the network based on a user's

[1]S. Chatterjea, S. de Luigi and P. Havinga. An Adaptive, Directed Query Dissemination Scheme for Wireless Sensor Networks. In *Proceedings of the Thirty-Fifth IEEE Conference on Parallel Processing Workshops* (ICPPW), August 2006, Columbus, USA. pp. 181-188, IEEE Computer Society Press.

injected query[2,3]. The results indicate how AI-LMAC efficiently manages the issues of fairness and latency.

[2]S. Chatterjea, L.F.W. van Hoesel and P. Havinga. A Framework for a Distributed and Adaptive Query Processing Engine for Wireless Sensor Networks. In *Transactions of the Society of Instrument and Control Engineers*, E-S-1 (1). pp. 58-67, 2006.

[3]S. Chatterjea, L.F.W. van Hoesel and P. Havinga. AI-LMAC: An adaptive, information-centric and lightweight MAC protocol for wireless sensor networks. In *Proceedings of the First IEEE Conference on Intelligent Sensors, Sensor Networks and Information Processing* (ISSNIP), December 2004, Melbourne, Australia, IEEE Computer Society Press.

4.1 Introduction

WSNs designed to service queries generally need to carry out three main tasks:

1. Disseminate the query into the network.
2. Sample the sensor specified in the query.
3. Return the acquired data to the sink that originally injected the query.

In this chapter we describe two different mechanisms. The first mechanism which addresses Task 1, describes a Directed Query dissemination scheme, DirQ, that routes queries to the appropriate source nodes based on both static and dynamic-valued attributes such as sensor types and sensor values. The second mechanism places an emphasis on Task 3 (though it actually also applies to Task 1). It describes an Adaptive, Information-centric and Lightweight MAC (AI-LMAC) protocol that adapts its operation depending on the amount of data traffic that is expected to flow through the network based on a user's injected query.

While both DirQ and AI-LMAC could generally be used in the same application (see Section 4.2), they are designed for slightly different scenarios. DirQ can be used to handle one-shot queries where the user only requires a single "snap-shot" of certain sensor readings. AI-LMAC on the other hand is used when data needs to stream in, in response to a particular query thus allowing the end-user to monitor parameters in close-to real time.

DirQ and AI-LMAC are ideal examples of how cross-layer information from the application layer can be used to influence the lower layers of the WSN protocol stack as described in Chapter 1. Both mechanisms use a similar approach in the sense that nodes use a framework to first locally classify the very data that is sensed. This classified information is subsequently presented to the routing and MAC layers so that they can operate more efficiently.

Before describing the specifics of DirQ or AI-LMAC, we first highlight the assumptions we make based on the application that motivates our research in this chapter. The DirQ scheme is presented in Section 4.3. Section 4.3.1 presents current research related to the area of query dissemination in WSNs. We then present the actual operation of DirQ in Section 4.3.2 and an analytical evaluation in Section 4.3.3. Section 4.3.4 describes details of the Adaptive Threshold Control mechanism which allows nodes to change update rates autonomously. Simulation results are presented in Section 4.3.5. AI-MAC is presented in Section 4.4. We first begin with current research related to the area of MAC

protocols designed specifically for wireless sensor networks. In Section 4.4.2 we describe how the data collection framework is developed to support the requirements of AI-LMAC. Section 4.4.3 explains how the MAC protocol adapts its operation using the data collection framework presented in Section 4.4.2. The results are presented in Section 4.4.4. Finally, the chapter is concluded in Section 4.5.

4.2 Assumptions made based on application

The work presented in this chapter is targeted for applications such as environmental monitoring where a large number of queries can be expected to be injected into the network. For example, the application mentioned in Section 2.3.3 describes a sensor network laid out to monitor various physical parameters in a forest. The WSN may be used by researchers at the university, students and the public in general. Based on this application, we have made a few assumptions about the queries posed to the network and about the network itself. Firstly, due to the large variety of users, ranging from researchers to students and even the public, the network can expect to be faced with a very large number of queries. Queries can also be multi-dimensional in nature, e.g. while some users might be interested in temperature data, others might be interested in humidity. Users are expected to inject one-shot range queries into the network, e.g. "Acquire all temperature readings that are currently between 22°C and 25°C." This also means that there could be some overlaps of certain parts of multiple queries (both in the spatial and temporal sense) that may be injected into the network at any single point of time. An implication would be that even though data may not be generated at very high rates (e.g. certain parameters may change very gradually) a large number of queries could be expected to be active within the network simultaneously. Also it may be possible that certain queries generate very high data rates in certain parts of the network while other parts can be relatively inactive. We also assume that the server connected to the root of the sensor network (which is in-charge of injecting queries) is capable of predicting the number of queries that will be posed to the network in the next hour based on historical data. The techniques used to make this prediction can be similar to the ones used in web servers to predict web page accesses [52].

In terms of the assumptions about the network, we assume a dense network where all node locations are fixed during deployment. The network consists of heterogeneous nodes where different nodes may have different sensors attached to them. Nodes (with even new sensor types) may be added to or removed from

the network at any time. The developed system should also be able to cope with node failures.

4.3 An Adaptive Directed Query Dissemination Scheme

This section focuses on how to query the environmental sensor network in an energy-efficient manner. Thus, our design describes how to accurately locate the necessary sensor nodes to help fulfill the request specified in the query injected by a user into the network. This section does not deal with the data extraction mechanisms that come into play once the required data sources have been identified and located. When designing a querying scheme for WSNs, it is important to recognize the various forms of dynamism that exist within the network due to both extrinsic and intrinsic factors. An example of an extrinsic factor would be the varying rate at which queries may be injected into the network while an intrinsic factor would be the rate of variation of the physical parameter being measured or changes in network topology. Rather than designing a static querying scheme which disregards all sources of dynamism, a protocol designer should try to take advantage of these variations by allowing the querying scheme to adapt accordingly. For example, nodes should spend more energy capturing information when user demand is high or when significant changes take place in the physical parameters being measured. Conversely, nodes should try to conserve energy by operating more economically during periods of inactivity. In this section we describe an adaptive directed query dissemination scheme (DirQ) which tries to ensure that queries are only delivered to the relevant nodes in the network. By relevant nodes, we refer to nodes that are able to service a particular query (source nodes) and also to forwarding nodes through which a query needs to propagate in order to reach the source nodes. The DirQ scheme works with directed dissemination of queries in contrast to the conventional approach of flooding the entire network every time a user poses a query. In order to ensure accurate delivery, all nodes in the network need to store some routing information that can eventually be used for limiting the dissemination of a query only to the appropriate regions of the network. Instead of updating this information on a periodic basis, DirQ adapts the update rate based on the rate at which queries are injected into the network and the rate of variation of the physical parameter being measured by the sensors. While DirQ is predominantly designed for fixed networks, it is able to cope with changes in network

topology caused by the addition or death or removal of sensor nodes. This is due to the fact that DirQ incorporates certain cross-layer features which allow it to gather information from the underlying MAC protocol [78]. It also has a scalable architecture as it allows for the addition of new sensor types after deployment of sensors has been completed, i.e. a user is not required to have prior information about all the types of sensors that may be added to the network after the initial deployment. Our simulations indicate that DirQ is fairly accurate (i.e. it suffers from a maximum overshoot of only around 2%), even though the cost of DirQ hovers between 45% and 55% of the cost of flooding. For more details we refer the reader to Sections 4.3.4 and 4.3.5.

4.3.1 Related work

Our work focuses on directing injected one-shot range queries to the relevant parts of the network instead of carrying out a simple flooding of the entire network. What sets DirQ apart from the majority of the other existing schemes for sensor networks is that queries can be directed based on a combination of static and dynamic attributes, e.g. sensor values (dynamic), sensor types (static) and even location (static) if it is available. Thus two identical queries could follow two completely different paths at different times of the day. Moreover, since DirQ can operate based on just sensor value and sensor type information, it is not dependent on any underlying localization mechanism. Having location information would of course extend the capabilities of DirQ. The problem of directing queries to the appropriate section of the network has been addressed in numerous query dissemination mechanisms. Directed diffusion [82, 81], is an example of a data-centric routing protocol, i.e. routing is performed based on the name of the data rather than the identity of the destination node. The Directed diffusion substrate can route queries to specific locations. However, it requires geographic information embedded in the query in order to do this. Our work is closely related to Semantic Routing Trees (SRT) presented in [96]. SRT is based on a distributed index. SRT however, only considers single attributes where as DirQ can use multiple attributes. Also, SRT is more suited for static attributes such as location, where as DirQ is capable of working with static and dynamic attributes. The authors in [147] present COUGAR which views the sensor network as a distributed database. However, the main problem with their approach is that a large amount of meta-data needs to be transferred to the query-processor on a periodic basis. The architecture of DirQ ensures that nodes are able to take autonomous decisions based on locally available information and thus adapt to changing network dynamics. There are also a

number of data-centric storage mechanisms designed specifically for wireless sensor networks. DCS [126] is a scheme which provides a hashing function for mapping an event name to a specific location. DIFS [72] and DIM [70] extend the data-centric approach to provide spatially distributed hierarchies of indexes to data. However, DCS, DIFS and DIM all require location information and are actually dependent on GPSR (Greedy Perimeter Stateless Routing, [85]) as the underlying routing protocol. Additionally, these complex indexing schemes may be too resource-hungry for sensor-class nodes as they require maintaining significant state information and large tables. It should be noted that DIM for example, was tested on PDA and PC platforms. The operation of DirQ on the other hand, has been kept relatively simple as nodes only store information from their local, one-hop neighborhood.

4.3.2 Operation of DirQ

The operation of DirQ has a number of phases. We first provide the reader with an overview of the phases and then describe each of them in greater detail in the following sections. Once the nodes have been placed in the network, a spanning tree is set up. Subsequently, the root node broadcasts a message with the estimate, E_{Hr}, of the number of queries that can be expected to be injected into the network over the next hour. This operation is repeated by the root periodically, e.g. once every hour. Next, a receiving node combines the estimate received with the local conditions of the physical parameter being measured (e.g. temperature, humidity, etc.) to decide upon a suitable threshold value as described Section 4.3.4. A node decides whether to transmit an update message to its parent node when the threshold value is exceeded. An update message is made up of a tuple that consists of minimum and maximum sensor readings and is used by nodes to maintain range information of sensor readings. This range information is subsequently used to direct a query only to the relevant nodes instead of flooding the entire network. Our approach of using minimum and maximum sensor readings has been adapted from earlier work presented on Semantic Routing Trees (SRTs) [96]. However, instead of using sensor readings, SRTs use only static location information.

4.3.2.1 Maintaining the range table

Upon receiving an E_{Hr} message from the root, a node calculates the threshold value, δ. (Section 4.3.4 describes how δ is calculated.) When a node acquires a sensor reading, R_{Aq}, it sets a minimum threshold, TH_{min} and a maximum

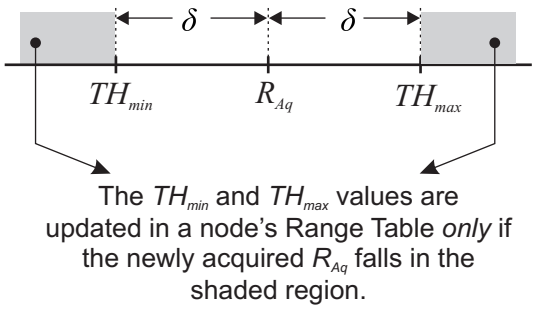

Figure 4.1: Local update of range table

threshold, TH_{max} where,

$$TH_{min} = R_{Aq} - \delta \tag{4.1}$$

$$TH_{max} = R_{Aq} + \delta \tag{4.2}$$

TH_{min} and TH_{max} are put together to form a tuple which is referred to as a *Range Message*. The Range Message is then inserted into the *Range Table* of the node. If a newly acquired sensor reading falls outside the range of TH_{min} and TH_{max}, it is considered as the new R_{Aq} and new TH_{min} and TH_{max} values are computed as shown above and inserted into the Range Table. However, if the newly acquired reading falls within TH_{min} and TH_{max}, the existing range values in the Range Table are left unchanged. Thus only major temperature changes are reflected in the node's Range Table. This process, that takes place within every node, is illustrated in Figure 4.1.

Apart from storing a node's own TH_{min} and TH_{max} values, the Range Table of a node also contains all the minimum and maximum threshold values of *all* its *immediate* child nodes (i.e. child nodes that are just one hop away). Thus the Range Table of a node that has n child nodes one hop away contains $n + 1$ tuples of TH_{min} and TH_{max} (i.e. one tuple for a node's own entry and one tuple each for all n child nodes).

Every time the Range Table of a node is modified (i.e. one of the TH_{min} or TH_{max} values is changed), the node parses through the table and picks out the minimum TH_{min} and the maximum TH_{max} among all the entries, i.e. $\min(TH_{min})$ and $\max(TH_{max})$ respectively. This is shown in Figure 4.2. These values are then compared with the previously transmitted $\min(TH_{min})$

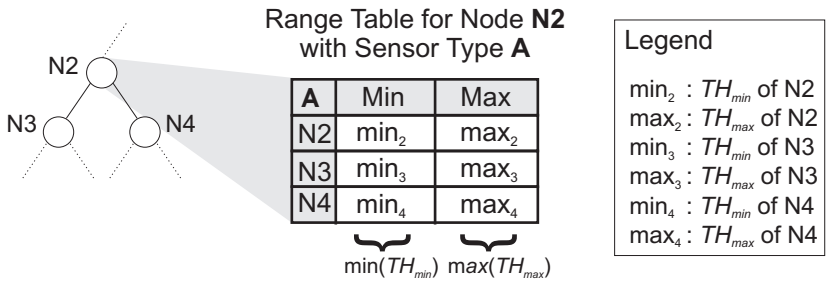

Figure 4.2: Picking the maximum and minimum threshold values

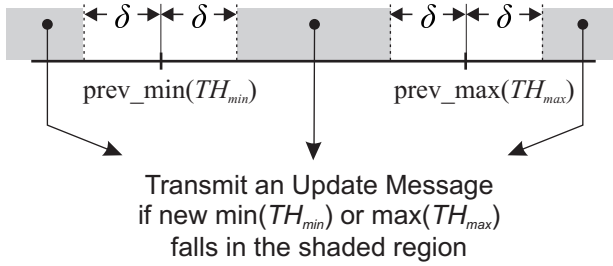

Figure 4.3: Transmission of an update message

and $\max(TH_{max})$ values, i.e. $\text{prev\_min}(TH_{min})$ and $\text{prev\_max}(TH_{max})$. If $\min(TH_{min})$ or $\max(TH_{max})$ differs from $\text{prev\_min}(TH_{min})$ or $\text{prev\_max}(TH_{max})$ respectively by an amount greater than δ, then the node transmits a new *Update Message* that contains the new $\min(TH_{min})$ and $\max(TH_{max})$ values to its parent. Thus, only significant changes in sensor readings trigger an Update Message that traverses upwards towards the root of the network which in turn ensures that the range information stored in nodes en route to the root remain updated. This helps to direct range queries more accurately to the relevant parts of the network. This process of transmitting Update Messages is illustrated in Figure 4.3.

Note that the δ used in Figures 4.1 and 4.3 have the same value. Let us temporarily assume that both values are different, i.e. Figure 4.1 uses δ_1 and Figure 4.3 uses δ_2. We now consider two separate cases and analyse the effects if:

- $\delta_1 > \delta_2$: In this scenario, a node will only update its own Range Table if

a newly acquired reading, R_N is such that $|R_N - R_{Aq}| > \delta_1$. However, since δ_2 is smaller than δ_1, effectively, a node will only transmit update messages when changes greater than δ_1 occur. In other words, δ_2 would not have any role to play.

- $\delta_2 > \delta_1$: In this scenario, a node would frequently update its own entry in its Range Table. But this would be a useless operation since the large δ_2 would prevent any Range Messages from being transmitted. Thus, δ_1 would not have any role to play.

The above reasons clearly indicate that it would be pointless to have two separate δs for Figures 4.1 and 4.3 thus justifying the use of just one value of δ in both cases.

A range query indicates the range of readings a user is interested in. Thus the range specified in the query is used in combination with the range information stored in the Range Tables of various nodes to disseminate queries to the relevant parts of the network. The general idea is to transmit a query to the child nodes which have ranges which overlap with the injected range query.

Every node can contain one or more Range Tables. Each Range Table contains range information for a single sensor type. As an example, Figure 4.4 shows that Node N1 maintains three separate Range Tables for sensor types A, B and C. This is despite the fact that N1 only has sensor type C. The presence of a particular range table within a node means that the corresponding sensor type exists either within the node itself, or within one or more of the children deeper within the tree structure.

Using multiple Range Tables and categorizing the incoming data according to ranges has several advantages. Firstly, it allows the network to be made up of heterogeneous nodes (i.e. different nodes can possess different combinations of sensors). This is a great benefit over previous architectures such as TinyDB which only supports homogeneous networks. Also, instead of flooding the entire network with queries, it would be possible to have a directed dissemination of queries. Thus the dissemination of queries could only be limited to certain sections of the network. The Range Table allows more complex query dissemination schemes to be derived from queries that are injected into the network by the end-user. These complex schemes could span across multiple Range Tables.

For example, the user could inject a query that requests for readings from sensor A only when the readings of sensor B and C are at a particular threshold. This would involve parsing the Range Tables of both sensors B and C. The Range Table also allows query optimization to be performed on an incoming

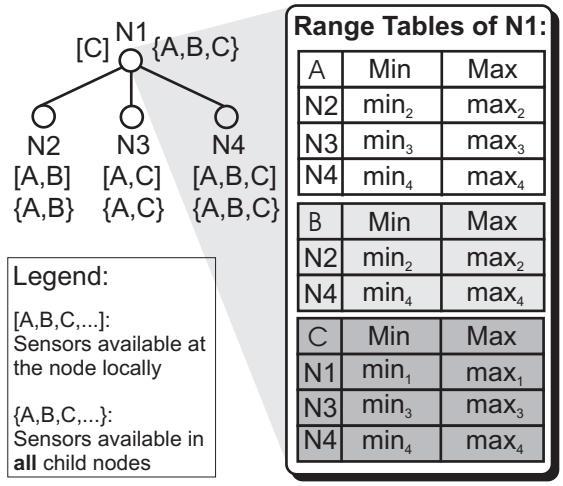

Figure 4.4: Support for multiple sensor types

query. Thus effectively, this system allows query optimization to be performed online (i.e. within the nodes) rather than centrally which is the traditional approach used in databases.

As can be seen from Figure 4.4, every Range Table includes a list of the immediate neighbours that have contributed to readings that fall within a certain range. Since the information contained in every Range Table of every node is purely based on the immediate neighbourhood, the size of the table is not dependent on factors such as the depth of the tree or network size. It is however, dependent on the number of different types of sensors present in the child nodes and the number of immediate child nodes a particular node has. Every row in the range table would have a size of 48 bits (Sensor type: 8 bits, Node ID: 16 bits, Minimum sensor reading: 12 bits, Maximum sensor reading: 12 bits). Thus if a node has up to 8 child nodes with a total of 5 different sensor types, this would mean that the Range Table would take up 40 rows or 240 bytes. Considering the fact that typical sensor node platforms have at least 4k of RAM (Table 2.1), a sensor node should not have any problems storing a Range Table.

4.3.2.2 Adapting to network dynamics

The Range Tables of DirQ are able to adapt to changes within the network topology, e.g. due to dead nodes or the addition of new nodes. This is because of DirQ's cross-layer interaction with LMAC [78]. When LMAC detects that a neighboring node has died, it sends a notification to DirQ which then checks to see how the removal of the neighboring node has affected its Range Table. Any changes in the range information are then propagated up the tree. Similarly, any changes in sensor types such as the addition or removal of sensors also propagate up the tree.

4.3.3 Analytical analysis

In this section we carry out an analytical comparison of the performance of flooding and directed query dissemination. Since the basic mechanism of DirQ is based on a tree, we perform our analysis using a k-ary tree with depth d. We also mention another reason for using a k-ary tree for the analytical comparison in Section 4.3.3.2. Additionally, the cost of transmitting a message is assumed to be one unit while the cost of receiving a message is also assumed to be one unit.

4.3.3.1 Cost of tree-based flooding

We first assume that a tree construction protocol is used to construct a tree rooted at the sink node. Using this protocol, every node knows the ID of its parent and also of all its immediate (i.e. one hop) children. Before describing how we analyze the cost of tree-based flooding (TBF), we first explain how a TBF operation is carried out in order to justify the analysis later on. We make the assumption that a tree building/maintenance protocol such as FixTree [106] is used to set up a communication tree. Using this protocol in combination with LMAC, every node would know which adjacent node is its parent node, which adjacent nodes are child nodes and which nodes are neither parent or child nodes. Thus when a node receives a message, it only processes the DM section 2.5.1 if the node that has broadcast the message is a child or parent node. If the transmitting node is not a child or parent (i.e. it is just another neighbouring node within range, e.g. N4 and N6 in Figure 4.5), the receiving node simply stops listening to the DM section of the incoming message.

To compute the total cost of the TBF scheme, which includes the cost of all the transmit (C_{Tx}) and receive (C_{Rx}) operations, we consider the following.

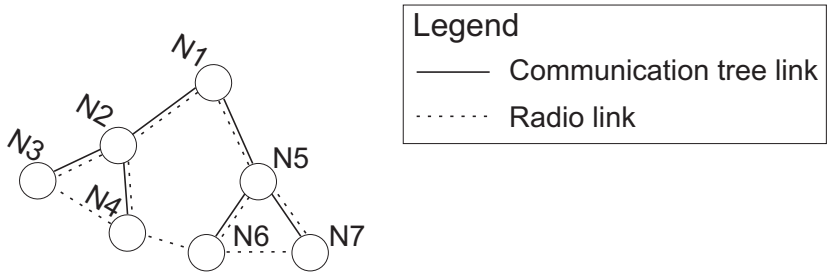

Figure 4.5: Nodes within radio range in a communication tree

During the flooding operation, every node in the tree, with the exception of the leaf nodes, broadcasts the query. This broadcast message is not processed by neighbouring nodes that are adjacent but are not a parent or child node of the transmitting node. The total cost of a flooding operation includes the cost of all the transmit and receive operations. Note that the number of nodes, N, in a k-ary tree with depth d is:

$$N = \frac{k^{d+1} - 1}{k - 1} \tag{4.3}$$

And the number of edges, E, is:

$$E = N - 1 = \frac{k^{d+1} - k}{k - 1} \tag{4.4}$$

The leaf nodes do not carry out a broadcast. Also we disregard all transmit/receive operations performed by the root since it is powered. Thus the total number of transmit operations in a k-ary tree with depth d is as follows:

$$C_{Tx} = \frac{k^d - 1}{k - 1} - 1 = \frac{k^d - k}{k - 1} \tag{4.5}$$

Recall we mentioned earlier that a node only processes the DM section of a received message if the node that sent the message is either a child or parent node. We also mentioned above that only broadcast operations are used. This implies that when a node carries out a broadcast operation, both the parent and child nodes receive the data. Thus every node in the network except the root and leaf nodes experience two "receive" operations. The leaf nodes experience

only one "receive" operation since they do not have any children. Thus, the total number of receive operations is:

$$C_{Rx} = 2 * (\frac{k^d - k}{k - 1}) - k + k^d \tag{4.6}$$

Thus the total cost of a TBF operation, CF_{Total}, for a k-ary tree with depth d, is the sum of Equations 4.5 and 4.6:

$$CF_{Total} = \frac{k^{d+1} + 2k^d - k^2 - 2k}{k - 1} \tag{4.7}$$

4.3.3.2 Cost of directed query dissemination scheme

In a flooding operation, energy is consumed disseminating a query to the entire network. In DirQ however, there are two sources of energy consumption - (i) directing a query to the relevant nodes in the network and (ii) the update mechanism to keep the minimum and maximum ranges updated. As there are two sources of energy consumption in the directed query dissemination scheme as opposed to only a single source in the flooding mechanism, the reader may be inclined to think that DirQ consumes more energy than the flooding mechanism. However, we ensure that this does not happen by adding an adaptive feature to the second source in DirQ, i.e. the update mechanism known as the *Adaptive Threshold Control* (ATC) mechanism. ATC ensures that the cost of DirQ always remains below the cost of flooding.

The cost of function of DirQ can be defined as follows:

$$CT_D = CQ_D + CU_D \tag{4.8}$$

where CT_D is the total cost of DirQ for a single query, CQ_D is the cost of disseminating one query to the relevant nodes and CU_D is the cost of sending an update message. It is important to realize that the cost of disseminating a query to the relevant nodes, i.e. CQ_D, is really dependent on where the nodes are located within the communication tree. For example, if the nodes relevant to the query are located close to the root, the dissemination cost will be much lower than another query whose relevant nodes might be located much deeper within the tree. This is because the deeper the relevant nodes are within the tree, the greater will be number of nodes involved in forwarding the query. The spread of the relevant nodes within the tree is also another factor that affects the cost of disseminating a particular query, i.e. the greater the spread of the relevant nodes, the greater the dissemination cost. From these examples, it is

apparent that the maximum cost of disseminating a query (i.e. CQ_{Dmax}) will occur only when all the leaf nodes of a tree are involved in servicing a query. This is another reason why we use a k-ary tree (as opposed to using an incomplete tree) to compute the maximum cost of query dissemination as this ensures that the number of leaf nodes is maximum.

The computation of the worst case of DirQ occurs when both CQ_D and CU_D are maximum. CU_{Dmax} occurs when all the nodes in the network transmit an Update Message.

Furthermore, it should be noted that when calculating the maximum query dissemination cost, the leaf nodes do not transmit the query. This implies that only nodes within $d-1$ hops from the root will be involved in query transmission. The maximum cost of query dissemination, CQ_{Dmax} is dependent on the total number of transmit ($CQ_{Dmax,Tx}$) and receive ($CQ_{Dmax,Rx}$) operations. Thus,

$$CQ_{Dmax} = CQ_{Dmax,Tx} + CQ_{Dmax,Rx} \tag{4.9}$$

Since all nodes except the root and leaf nodes carry out one broadcast operation,

$$CQ_{Dmax,Tx} = \frac{k^d - 1}{k - 1} - 1 = \frac{k^d - k}{k - 1} \tag{4.10}$$

The tree maintenance protocol enables a node to know its parent and child node, i.e. the IDs of the parent/child are known. Using LMAC a receiving node knows the ID of the sending node. A node would then know if the message has originated from a parent or child. Thus in this case, the DM section is not received if the message has come from a child node. In general, broadcast messages from child nodes are ignored.

Since all nodes except the root carry out one receive operation,

$$CQ_{Dmax,Rx} = N - 1 = \frac{k^{d+1} - k}{k - 1} \tag{4.11}$$

Thus the maximum cost of query dissemination, CQ_{Dmax} is the sum of Equations 4.10 and 4.11:

$$CQ_{Dmax} = \frac{k^{d+1} + k^d - 2k}{k - 1} \tag{4.12}$$

For calculating CU_{Dmax} we assume that all the nodes in the network send one update message. Note that the update messages are aggregated as they propagate upwards towards the root. Thus a node combines all the messages

it receives from its immediate child nodes and transmits only one aggregated update message. Note that messages are transmitted using unicast operations. Since the root node is not counted, the number of transmit operations is equal to the number of edges in the tree:

$$CU_{Dmax,Tx} = \frac{(k^{d+1} - k)}{k - 1} \tag{4.13}$$

Similarly, since we do not count the root's receive operations, the number of receive operations is:

$$CU_{Dmax,Rx} = \frac{(k^{d+1} - k)}{k - 1} - k \tag{4.14}$$

Thus the maximum cost of sending an update is the sum of Equations 4.13 and 4.14:

$$CU_{Dmax} = \frac{2k^{d+1} - k^2 - k}{k - 1} \tag{4.15}$$

We introduce a variable f which indicates the frequency at which updates are received at the root. So for example, if one update is received at the root every 10 queries, then $f = 0.1$. Then the maximum cost of DirQ is,

$$CT_{Dmax} = CQ_{max} + f(CU_{Dmax}) = \frac{k^{d+1} + k^d - 2k}{k - 1} + f(\frac{2k^{d+1} - k^2 - k}{k - 1}) \tag{4.16}$$

4.3.3.3 Keeping the cost of DirQ below that of flooding

When using DirQ, it is essential to ensure that its cost is always kept below that of flooding. Thus we need to consider the worst case of DirQ and adapt it to make sure that it does not exceed the cost of flooding. This is performed by ensuring that $CT_{Dmax} < CF_{Total}$. Thus,

$$f_{Max} < \frac{k^d - k^2}{2k^{d+1} - k^2 - k} \tag{4.17}$$

where f_{Max} is the maximum number of updates that a node can transmit per query to guarantee that DirQ does not surpass the cost of flooding. So as an example, if k = 2 and d = 4, then $f_{Max} < 0.21$, i.e there can be at the most 0.21 updates per query or 1 update every 4.8 queries in order for the directed query

dissemination scheme to be more energy efficient than flooding. Note however, that this is for the worst case. This implies for example that you could have say 0.8 updates per query (i.e. $f_{Max} = 0.8$) when the query is disseminated only to a few nodes that are close to the root or all nodes in the network are not involved in sending update messages.

4.3.3.4 Keeping k and d values updated

It is evident from the discussion above that a node needs to know the values of the variables k and d whenever it needs to compute f_{Max}. The values of these variables are computed by the nodes and the sink during initialization when the communication tree is being built. In general, every node transmits the number of immediate children it has unless one of its child nodes has a larger number of children than itself. In such cases, the node forwards this larger number to the sink. Thus ultimately, the sink gets to know the maximum number of children that a particular node in the network has. Similarly, every node transmits its distance from the gateway unless it has a child node that has a greater distance than itself. In this case, the parent node forwards the larger distance to the sink. Subsequently, the maximum number of children, k and maximum distance, d are broadcast to the entire network.

After initialization has been completed, the nodes also retransmit values of k and d to the sink when changes in topology are detected. Any node which detects a greater number of children than k informs the sink of the change. Similarly, any node originally with k children having a fewer number of children (e.g. due to the death/removal of a node) also informs the sink. A similar mechanism is used to keep d updated as well.

4.3.4 Adaptive threshold control

In Section 4.3.2 we described how Update Messages are transmitted only when the new minimum or maximum threshold values exceed the threshold δ. The precise value of δ is not hard-coded into DirQ and neither is it predefined and fixed by the user. Instead, every node decides for itself what the threshold value should be. The decision is based on how heavily the network is being utilized (in terms of the rate of injection of queries) and the degree of variability of the physical parameter that is being measured by a sensor type.

DirQ uses an Adaptive Threshold Control (ATC) mechanism which changes the value of δ dynamically. The chosen value of δ is dependent on the number

of queries that are expected to be injected into the network over the next hour and also on the rate of change of the measured data.

We assume that the root node that is in-charge of injecting a query into the network is capable of making predictions of the number of queries that will be posed to the network during any particular hour of the day. The technique used to do this can be similar to those used by current web servers which are able to predict incoming hits to a particular web site based on historical data. This estimate, Q_E, is then sent by the root node to all the nodes in the network.

Recall how we illustrated in Equation 4.16 that the maximum cost of DirQ is dependent on the number of updates received at the sink per injected query. Thus once a node computes f_{Max} the node can use f_{Max} in combination with the estimated number of injected queries, Q_E, to compute the number of Update Messages that it can transmit over the next hour, $U_{Max/Hr}$, using,

$$U_{Max/Hr} = Q_E \times f_{max} \tag{4.18}$$

Note that $U_{Max/Hr}$ is the maximum number of Update Messages that can be transmitted per hour given the current hourly estimate of the number of injected queries to ensure that DirQ does not surpass the cost of TBF.

A node uses then uses the computed $U_{Max/Hr}$ to calculate the maximum number of Update Messages that can be transmitted over the next 10 minutes, i.e. $U_{Max/10mins}$.

The next part of the problem is to see how δ can be locally adjusted by every node to ensure that the rate of transmitting Update Messages follows $U_{Max/Hr}$ closely. A node initially uses a predefined (usually large) value of δ to start off with. After the first 10 minutes, the node checks the number of Update Messages that were transmitted. If it is less than $x_1\% \times U_{Max/10mins}$, the value of δ is reduced by α. However, the value of δ never reduces beyond a user-defined δ_{min}. This ensures that unnecessary Update Messages are not transmitted when the sampled sensor readings are relatively constant.

Conversely, if the number of transmitted Update Messages is greater than $x_2\% \times U_{Max/10mins}$, the value of δ is increased by α. This process of adjusting δ is carried out every ten minutes to ensure that the number of transmitted Update Messages lies between $x_1\%$ and $x_2\%$ of the TBF scenario. The greater the value of α, the greater is the impact in the change of the number of update messages transmitted in every ten minute block. However, we do not investigate the precise effect of the value of α in this thesis and leave this for future work.

In our simulations described in Section 4.3.5, we set $x_1\%$ and $x_2\%$ to 45% and 55% respectively. Generally, the smaller the values of $x_1\%$ and $x_2\%$, the lesser

will be the number of transmitted Update Messages. This will lead to more inaccurate routing of range queries due to the less accurate range information stored in the nodes. Conversely, larger values of $x_1\%$ and $x_2\%$ will lead to more accurate routing at the cost of increased transmission of Update Messages. The value of δ_{min} is set to 1 unit and α is set to 0.2 units.

4.3.5 Simulation results

In this section, we evaluate the performance of DirQ using simulations and compare its performance in relation to tree-based flooding (TBF). Our simulations intend to highlight the strengths of DirQ in terms of energy savings and also in terms of the accuracy of our directed query dissemination scheme. The primary objective of the simulations is to illustrate how DirQ maintains a cost below that of flooding and yet attains a high level of accuracy for the current network and environmental conditions completely autonomously. We initially examine the effects of accuracy and efficiency of query dissemination when using fixed thresholds (i.e. fixed values of δ). We then illustrate how the ATC mechanism curbs the total cost such that it is below flooding and yet maintains an acceptable level of accuracy. The simulation is performed using OMNeT++ which is a discrete event simulator [15]. The results are based on a network topology of 50 nodes which includes one root where $k = 8$ and $d = 10$. DirQ was implemented on top of the LMAC protocol. A synthetic dataset with 4 sensor types has been generated (with sensor values 0 to 100 units) where sensor values of nodes located close to one another are spatially related. The generated sensor data is also related in the temporal dimension. Each sensor node acquires a reading every time unit for a period of 20,000 time units. We refer to each time unit as an epoch. Random queries which covered 20%, 40% and 60% of the nodes are generated every 20 epochs.

4.3.5.1 Using fixed threshold values

We initially perform simulations to investigate the effects when fixed threshold values were used. Threshold values are fixed at δ= 0.5 units, 1 unit and 3 units. For every value of δ we also examined how the percentage of nodes involved in responding to a query would affect the results. Note that the percentage of nodes involved in a query is not directly dependent on the selectivity of the query itself. As an example, even if the selectivity of a particular query is very high (i.e. only a small number of nodes are involved) the percentage of nodes involved in answering the query is highly dependent on the location of the

relevant nodes within the communication tree. If the relevant nodes are located very close to the root, the selectivity would be proportional to the number of nodes involved in the query. However, if the nodes are located deep within the network, propagating the query to the relevant nodes would be a lot more expensive due to the large number of intermediate forwarding nodes involved. Thus our definition of "percentage of nodes involved in responding to a query" involves not only the relevant nodes but also the intermediate forwarding nodes.

Our first result studies how the accuracy of the directed query dissemination scheme is affected using various levels of δ. Let us first state how we define the term accuracy. When a query is injected, ideally it should be directed only to the relevant nodes in the network. However, this does not happen in reality due to the threshold levels used in the network. Since the transmission of updates is dependent on the value of δ and the rate of change of the measured parameters, the range values maintained by the nodes is not always accurate. Thus in certain instances, queries could be routed to nodes which are not relevant to a particular query. Naturally, routing queries to the non-relevant nodes, also consumes energy. We measure accuracy by computing the proportion of nodes that are being reached in response to a query to nodes that should be reached. Nodes that "should" be reached refer to both source nodes and intermediate forwarding nodes.

The results in Figure 4.6 indicate that as the threshold increases (i.e. value of δ) the difference between the percentage of nodes that receive a query and the percentage of nodes that should receive the query increases. This is because as δ increases, the range information becomes more inaccurate. This effect is less pronounced as the percentage of relevant nodes increases. This is because when more nodes are involved in servicing a particular query, the probability of routing queries to wrong nodes diminishes greatly. The value of δ clearly plays a more significant role for queries with high selectivity.

4.3.5.2 Using the adaptive threshold control

We now describe the effects of the Adaptive Threshold Control (ATC) scheme. ATC enables individual nodes to autonomously adjust the value of δ. The main drawback of using a fixed threshold is that there is a possibility that the cost of the directed dissemination scheme may exceed the cost of TBF. Having a fixed threshold makes the system completely dependent on the parameter being measured. Figure 4.7 shows the total number of Update Messages that are transmitted by all the nodes in the network every 100 epochs over a period of 20,000 epochs. It can be seen that the ATC is successfully able to adapt the

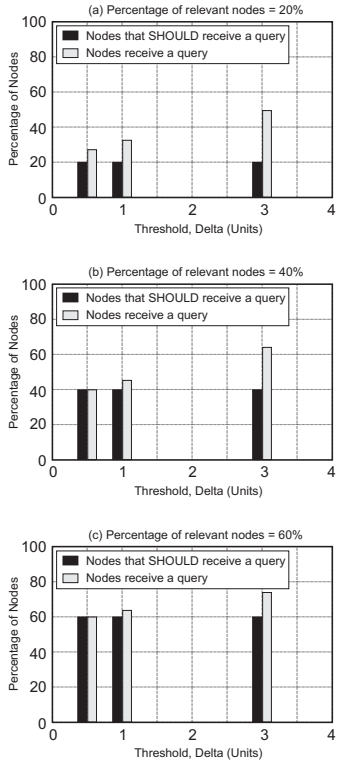

Figure 4.6: Effect of delta on accuracy (Percentage of relevant nodes: (a)20%, (b)40%, (c)60%)

transmission rate such that its cost lies around the region where the cost is roughly around 45-55% the cost of TBF.

Figure 4.6 has shown that the danger of increasing δ results in the query dissemination scheme becoming less accurate especially for queries with high selectivity. The main idea of having the ATC is to ensure that while the number of updates transmitted is limited, the accuracy (i.e. overshoot) should not decrease significantly. Figure 4.8 shows that the maximum overshoot when using the ATC is only around 2%. Figure 4.9 shows that the ATC reduces the total number of messages transmitted. It can be seen that while the ATC transmits a larger number of updates than when $\delta = 3$ units, the $\delta = 3$ scenario

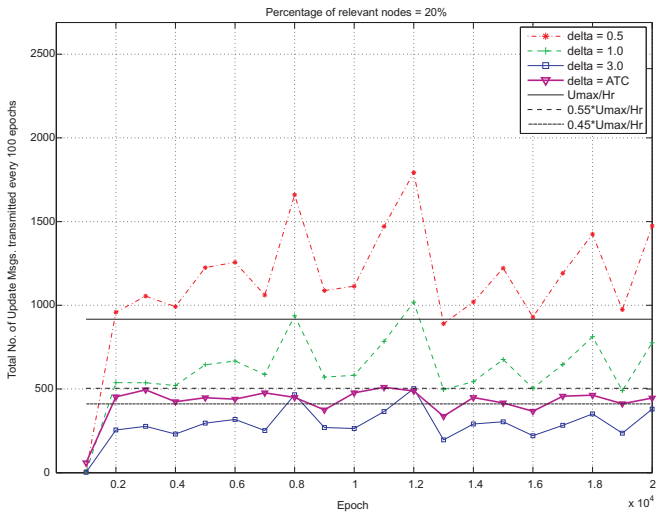

Figure 4.7: Number of update messages transmitted over time using different δ and the ATC

results in a greater number of wrong transmission of queries.

4.4 An Adaptive, Information-centric and Lightweight MAC Protocol for Wireless Sensor Networks

As mentioned earlier in Chapter 1, WSNs are typically application specific networks. Thus it is important to investigate the possibility of designing communication protocols that are specifically designed for a particular application in order to improve its level of efficiency. In other words, a protocol should be able to take advantage of certain inherent behavioural properties of the application being considered.

Judging from the previous statement, the reader may be inclined to think that following such a strict "application-specific" approach would result in a protocol that is completely static and is unable to adapt to any changes. We would like to emphasize, however, that we believe it is very important to design a protocol that is dynamic. But we also feel that instead of designing a

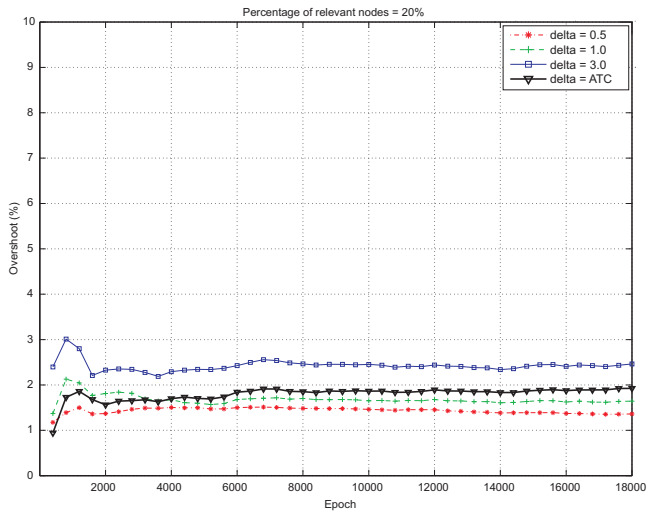

Figure 4.8: Overshoot using different δs and the ATC

protocol that is able to adapt from one application to another, the developed protocol should be able to adapt within the constraints of the application being considered.

In this section we describe the design of an adaptive, information-centric and lightweight medium access control (AI-LMAC) protocol for wireless sensor networks (WSNs) that is based on the Lightweight Medium Access Protocol (LMAC) [78]. This is very different from other existing MAC protocols for WSNs [78, 148, 59, 92, 146, 124] that operate independently of the queries injected into the network or the kind of data that flows within the network. However, apart from just describing the details of a new MAC protocol we also describe the design of a *Data Distribution Table* that resides on every node. This framework helps to capture data about the data flowing through the network. In other words, it captures network metadata. The information captured by the framework is then utilised by the MAC protocol to adapt its operation accordingly.

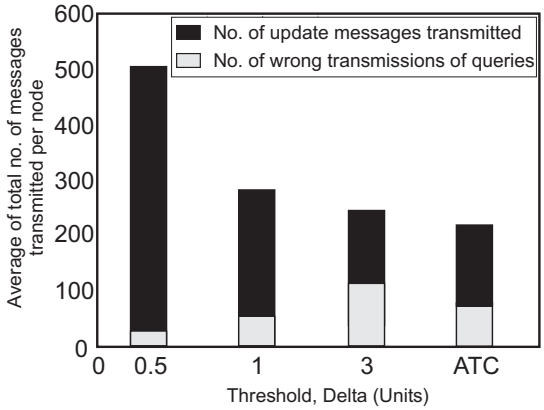

Figure 4.9: Net effect of threshold on the number of transmissions

4.4.1 Related work

There have been numerous MAC protocols developed for WSNs over the past few years [78, 148, 59, 92, 146, 124]. We only highlight some of the more prominent protocols here due to space limitation. Sensor-MAC (SMAC) [148] reduces energy consumption by reducing the duty cycle of a sensor node. This is done by following a periodic active/sleep schedule that is fixed. Nodes turn off their radio during sleep periods and turn it on while receiving or transmitting. As SMAC uses a fixed duty cycle, it is unable to adapt its operation to varying traffic rates. Timeout-MAC (TMAC) [59] improves on SMAC by using an adaptive duty cycle thus adapting automatically to traffic fluctuations. However, due to its aggressive power-down policy, nodes often go to sleep too early, thus decreasing throughput and increasing latency [88]. Data-gathering MAC (DMAC) [92] also uses an adaptive duty cycle where the wake up schedule depends on the depth of a node in the data gathering tree. Additionally, it provides a low source-to-sink latency. However, it does not take fairness into account. Thus the duty cycle assigned to a certain child node may not be proportional to the amount of data it needs to transmit when compared to another child of its parent node, i.e. a sibling node. [146] does take fairness into consideration by introducing a rate control mechanism. It defines fairness as giving every node in the network an equal opportunity to transmit its data. We, however, are not interested in having all nodes transmit data simultaneously. Since we are considering a heterogeneous network where queries injected will be distributed

both in the spatial and temporal sense, we define fairness as follows: "Only nodes which are able to service an incoming query or nodes which have children which can service a particular query should be given the chance to transmit their data. Nodes which are not involved should be given lower priority. Thus the priority given to a node is directly proportional to the amount of data a node is expected to transmit." This is only a general definition of how we interpret fairness. More specifically, our mechanism uses what is known as "2-Dimensional" fairness. This is explained in Section 4.4.3. Unlike [146] which uses the analogy of "metering traffic onto a freeway where each node generating data is like cars trying to enter", our mechanism on the other hand dynamically changes the "number of lanes" in the freeway to accommodate varying data traffic rates. Like AI-LMAC, the traffic-adaptive medium access protocol (TRAMA) [124] uses a TDMA-based communication scheme that is able to adapt to actual traffic conditions. However, although TRAMA achieves high channel utilisation, it does so at the expense of considerable latency and high algorithmic complexity. It also fails to address the issue of fairness.

While some of the protocols mentioned above are able to adapt their operation to varying data traffic rates, none of them makes use of any knowledge of the actual application they are being used for. In other words the MAC layer is completely independent of the application running above it. Recently, however, members of the database research community have realised the importance of influencing the MAC protocol using information from queries injected into the network. TAG [97] for instance performs some sort of communication scheduling using inputs from the query that helps to reduce the burden placed on the underlying MAC. The scheduling mechanism decides when the MAC should be operational, i.e. listening or transmitting. At other times the radio is put to sleep. [150] describes a Data Transmission Algebra (DTA) that uses query scheduling to reduce collisions at the MAC layer. However, in the event that the schedule is unable to avoid collisions, they are handled at the MAC layer.

In both instances, there is a scheduling mechanism based on an incoming query or data that decides when to turn the MAC on or off. The MAC continues its normal operation during its active period, e.g. it could still be prone to collisions. However, the likelihood of collisions occurring is reduced. While we have encountered communication schemes which basically only turn the MAC on or off, we have not encountered any schemes in the current literature that will actually modify the operation of the MAC itself.

We improve on current cross-layer optimization methods by going a step further than simply switching the MAC on or off by allowing the MAC itself

to adapt its own operation depending on the query and the amount of network traffic within the network. The amount of expected network traffic is computed using a *Data Distribution Table* (DDT) which is stored locally in every node in the network. We describe how the DDT operates in the following sub-section.

4.4.2 Description of the Data Distribution Table

The Data Distribution Table or DDT is built into every node. It helps make deductions about the amount of data traffic that can be expected to flow through the network depending on the injected query and the distribution of the data that is being generated by the various sensor nodes. The information provided by the DDT helps to ensure that bandwidth is only allocated primarily to nodes which meet the criteria specified in the query and are thus expected to generate more traffic.

Every node maintains a DDT for every sensor type that is present in its entire subtree. This is a characteristic that is similar to DirQ's Range Table previously mentioned in Section 4.3.2.1. A node keeps information about its own sensor readings in the first row of its own DDT. It also maintains additional rows corresponding to every one of its immediate child nodes. So for example, if a particular node has 4 immediate child nodes, the node would have a total of 5 rows in its DDT. The contents of these additional rows depends on what has been transmitted by the respective child nodes.

All the rows of a DDT are subdivided into a pre-defined number of columns. Each column represents a particular range of sensor readings. Note that the ranges do not overlap and we assume that they are either defined by the user prior to deployment based on typical values that may be expected from the region being monitored and the degree of selectivity of injected queries or they may also be broadcast to the entire network as and when required. The range of readings represented by each column need not be fixed. For example, if we assume that the readings follow a normal distribution, the user may opt to have narrow ranges for the middle of the curve and wide ranges for the two tail ends. Such a scheme would also be useful if the majority of injected queries are highly selective over a narrow range of readings that occur more frequently. While a large number of ranges would result in more frequent transmission of update messages, this is not considered to be a significant disadvantage as the primary aim of AI-LMAC is to minimise the impact of data loss due to dropped messages and latency. Figure 4.10 summarizes how the DDT functions.

When a node samples its sensor, it checks which range the acquired reading falls in and marks the appropriate column in the first row of its own DDT with a

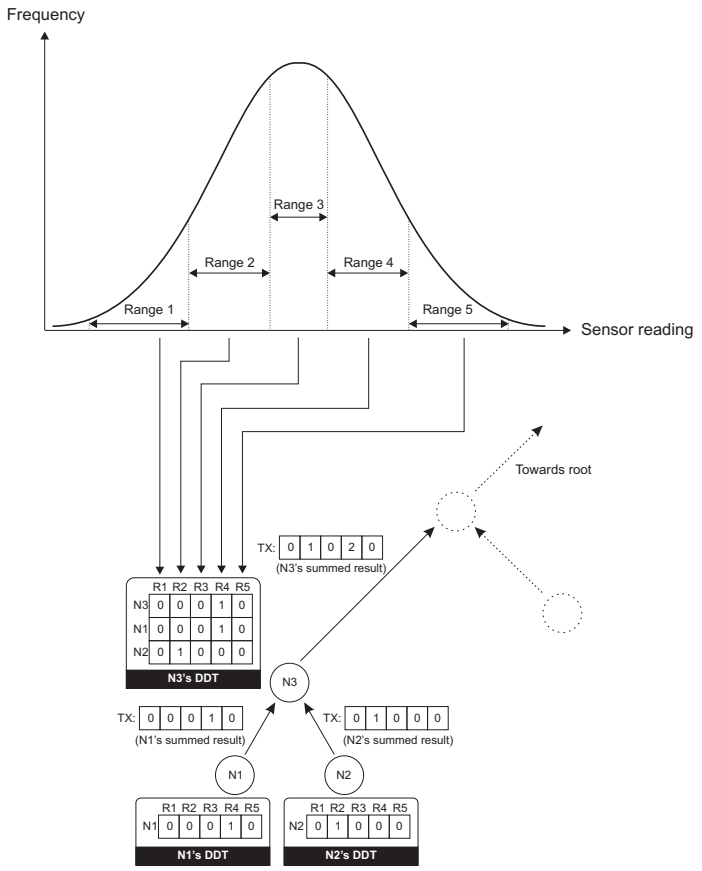

Figure 4.10: How the DDT functions

1. It then sums up every column of its DDT individually, resulting in a row and transmits this summed result to its parent node. Thus effectively, the summed result represents the number of children the particular node has in a certain range of readings *within its entire subtree*. If at any time a node detects x (e.g. $x = 5$) consecutive readings that fall within a different range from its previously transmitted range, it subtracts 1 from the column for the previous range and adds a one to the column representing the new range. The node then once again sums up every column of its DDT individually and transmits the result

to its parent node. Note that we use x consecutive readings and not a single reading so as to prevent frequent transmissions when readings oscillate about the border of two ranges. x is a user-defined parameter. A node transmits an updated summed result of its DDT when:

- its own newly acquired reading falls within a range that is different from the previously transmitted range,
- it receives a new summed DDT result from a particular child node,
- when a topology change is detected, e.g. a particular child node dies/ disappears or is added to the network.

Figure 4.10 provides an overview of how the DDT functions. Once again, as with the Range Table of DirQ, the size of the DDT is not dependent on the number of nodes in the network since it only consists of summed values. Instead, it depends on the number of different types of sensors used in the subtree of a node, the number of immediate children a node has and the number of ranges that the user wishes to use.

An additional advantage of the DDT is that when a node needs to decide how much bandwidth to allocate to each of its immediate child nodes, the decision is based on the *net requirement of all the queries* that are being executed by the node at that point of time. Thus effectively this means that the DDT allows query optimization to be performed in a distributed manner.

Note that although in this section we are emphasizing on how the DDT can be used to influence the operation of the MAC, it is evident that the DDT can also be used to direct range queries to the appropriate parts of the network due to the range information built into the DDT.

4.4.3 Adapting AI-LMAC using the DDT

We now describe how AI-LMAC can adapt its operation using the information provided in the DDT. Unlike LMAC, which allows every node within the network to own only one slot [78], AI-LMAC allows a node to own multiple slots. Also, AI-LMAC is able to vary the number of slots a particular node owns depending on the amount of data that is expected to flow through it. This ensures fairness in the sense that the bandwidth allocated to a node corresponds to the traffic it is expected to encounter. For example, it would be pointless to allocate a large number of slots to a particular node that is not generating or relaying significant amounts of data.

In AI-LMAC, we assume that a parent-child relationship exists between all the nodes in the network, such that the root of the network can be considered to be the highest parent in the hierarchy. Using the DDT, every node would know how much "importance" to give every one of its immediate children.

Using the DDTs a node cannot decide by itself, how much importance it should give itself to transmit. This is because the DDTs only contain information about a node's child information. A node is not aware of the data generated by the other children of its own parent node as they may not be in range. Thus, the parent is the only node that has knowledge of the proportion of data that will be contributed by each of its immediate children. The idea here is that if a node realises that a subset of its immediate children is going to transmit large quantities of data, then more attention needs to be paid to this particular subset of child nodes. In this case, when we say more attention, we actually refer to assigning multiple slots to a particular child.

However, even though a parent node knows which child node deserves more slots to be assigned to it, it cannot send such a rigid instruction to its children as in LMAC. This is because in LMAC, when a node performs slot assignment, it has knowledge of the slot ownership of its first and second order nodes. In this case, the parent node would not know slot ownership information about the slot assignments of its child node's second order nodes since they are three hops away.

Thus, the responsibility of the parent node is simply to "advise" the child, i.e. the parent node sends a message to every one of its children indicating the ideal number of slots that a particular node should take up under the current conditions. It is then up to the child node to follow the advice as closely as possible. This naturally depends on the number of empty slots available.

The process of giving advice starts at the root node of the tree when a query is first injected into the network. This process then percolates down the branches of the tree towards the leaf nodes. If, however, the process of giving advice started at an intermediate node, this would increase the chance of performing unfair slot allocations. This is because a node assigning slots would not be aware of the bandwidth requirements of all its sibling nodes which are not within its direct range. From this argument, it is obvious that if we apply this rule repeatedly, the root node is the only node which can assign slots fairly at the beginning. We term this as *horizontal fairness* as the mechanism ensures that all sibling nodes (i.e. a the same level) under a certain parent are allocated slots fairly.

Apart from establishing a horizontal relationship between nodes, we also introduce a mechanism to include *vertical fairness*. In order to prevent buffer

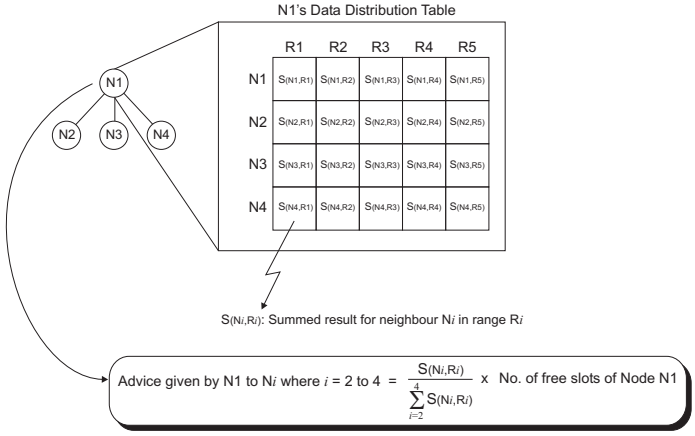

Figure 4.11: How the DDT is used to give advice

overflow problems, our mechanism ensures that that the total number of slots assigned to the immediate children of a certain parent node, does not exceed the number of slots owned by the parent. This reduces the likelihood of data packets being dropped due to lack of bandwidth. Furthermore, leaf nodes are prevented from being allocated excessive bandwidth using this mechanism.

Thus introducing *two dimensional fairness* ensures that the number of slots taken up by a node does not only depend on its siblings but on its parent as well.

Once a node has received the ideal number of slots it should take up, it checks to see which slots are free within its 2nd order neighbourhood following the normal protocol rules of LMAC. To ensure a balanced slot allocation between children nodes, nodes increment the number of controlled slots in turns at a rate of one time slot per frame. Just like in LMAC, once a node decides to take up a certain slot, it "marks" the slot using a "1" to indicate that the slot has been taken up. Figure 4.11 illustrates how a node uses its DDT to advise its child nodes on the number of slots to occupy.

4.4.4 Experimental Analysis

Our framework provides a mechanism to assign more bandwidth to those parts in the network that encounter more data traffic than others. In fact, the assigned bandwidth is proportional to the expected traffic. Hence our framework is able

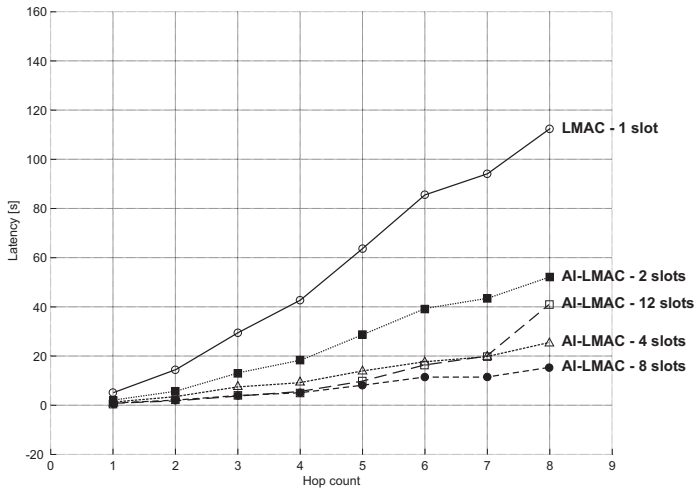

Figure 4.12: Comparison of LMAC (single slot allocation) with AI-LMAC (multiple slot allocation). Note that the maximum advice scenario of 16 time slots is not plotted as it matches the 12 time slots scenario)

to minimise the overall *latency* in the network and also the number of messages which need to be buffered in the nodes.

In our simulation, the DDT is not implemented. Instead, every node uses the total number of child nodes it has as an input parameter for computing the advice. When all nodes are active (i.e. have at least one time slot), the gateway node starts providing advice to its immediate children. The children follow the advice as closely as possible and create advice for their children as described earlier. Every node generates one message per four MAC frames. Thus the expected data volume is proportional to the the total number of child nodes that a node has.

Figure 4.12 compares the results for LMAC (single slot allocation) with AI-LMAC (multiple-slot allocation). These results are obtained by simulation using the discrete event simulator OMNeT++ [15]. The results are averaged over five different network topologies consisting of 49 nodes and one root. Ten different runs were carried out per topology. In AI-LMAC, the advice for the maximum number of allowable time slots is varied from 2 to 16.

The results clearly indicate that latency is proportionally reduced with the the maximum number of controlled slots (Figure 4.12). However, this holds

Table 4.1: Maximum number of back logged messages (worst case)

Scenario	Maximum messages
1 slot	105
2 slots	63
4 slots	34
8 slots	32
12 slots	54
16 slots	56

true only until eight slots. For the 12 and 16 slot scenarios, the number of free slots in the network rapidly decreases with every hop from the root and thus the nodes are not able to comply with the advice. Consequently, a bottleneck is created at a few hops (6 to 8) from the root, resulting in higher latency for messages created in those areas.

In this MAC protocol, nodes are able to receive up to 32 (=number of time slots) messages per frame. The number of messages that can be transmitted per frame is dependent on the number of time slots the node controls. When the number of incoming messages exceeds the number of messages that can be transmitted during a frame, the additional incoming messages have to be buffered. For each of the scenarios, we have collected data about the maximum number of messages back logged in the worst case (Table 4.1). These results reflect the same trend as Figure 4.12. Note that in real-life implementations, the capacity to hold back logged messages will be limited due to scarce memory resources in the sensor nodes.

4.5 Conclusion

In this chapter we present two mechanisms that use sensor readings obtained by the application layer to enable both the routing and MAC layers to operate more efficiently. The first mechanism deals with a Directed Query Dissemination Scheme (DirQ) which tries to ensure that queries injected into the network are only sent to the relevant nodes instead of flooding. DirQ routes queries based on sensor values and sensor types. It ensures that information in routing tables is accurate by sending update messages. The rate at which update messages are

transmitted is dependent on the level of usage of the network and the also the rate of variation of the physical parameter being measured by the sensors. Thus every node in the network is able to control its threshold level autonomously. The results indicate that DirQ spends between 45% and 55% of the cost of tree-based flooding.

The analysis we have presented for DirQ is based on the worst-case scenario. This approach has motivated us to assume that the k-ary tree is balanced. However, every query may not necessarily involve all the leaf nodes. Thus, it may be necessary to devise an additional off-line mechanism that monitors the pattern of injected queries. This can then be used to compute an average-case "f_{max}" rather than the worst-case which can subsequently be injected into the network. Nodes could then use this value to transmit updates more frequently. Thus while additional work would need to be carried out to monitor querying patterns centrally, the main mechanism of DirQ that runs within the network would remain the same.

The Range Tables in DirQ are designed based on ranges (i.e. minimum and maximum values) of sensor readings. There were several reasons for this design option. Firstly, the injected range queries, always specify a minimum and maximum value. These values could then be easily compared with the ranges stored in the Range Tables. Secondly, a node does not have to carry out extensive additional computation in order to identify the minimum and maximum values. Finally, the idea of monitoring ranges was initiated by the authors in [96] and the work presented in this chapter builds up on that previous work. It might be interesting to investigate if other parameters, e.g. variance could be monitored instead. However, this would involve more computation and the design would definitely be a lot more complex. Further investigation is required to see if such alternative design strategies would indeed provide better results.

As mentioned earlier, we have not investigated the effects of using different values of α in this chapter. Future work should include using different types of data sets (e.g. with different levels of variability - see Chapter 6) to see how the performance of DirQ varies when different αs are used with different data sets.

The second mechanism in this chapter describes a novel MAC protocol that adapts its operation depending on the type of query injected into the network. Thus unlike other conventional MAC protocols which run independently of the application, AI-LMAC is application dependent. We have also described a data distribution table that helps gather data about the queries and data flowing through the network and how it is subsequently used by the MAC layer to adjust its operation. Using a mechanism which depends on two-dimensional

fairness, we have illustrated how AI-LMAC reduces latency and handles the issue of fairness. Thus only parts of the network that need to respond to a certain query increase their level of activity. Other sections of the network remain relatively inactive. Our results show how AI-LMAC efficiently manages the issues of fairness and latency.

Chapter 5

A Distributed and Self-Organizing Scheduling Algorithm for Energy-Efficient Data Aggregation

In many environmental monitoring applications, environmental scientists are interested in collecting raw data using long-running queries injected into a WSN for analyzing at a later stage rather than injecting snap-shot queries into the network that contain data-reducing operators (e.g. MIN, MAX, AVG) that aggregate data. Collection of raw data poses a challenge to WSNs as very large amounts of data need to be transported through the network. This not only leads to high levels of energy consumption and thus diminished network lifetime but also results in poor data quality as much of the data may be lost due to the limited bandwidth of present-day sensor nodes. We alleviate this problem by allowing certain nodes in the network to aggregate data by taking advantage of spatial and temporal correlations of various physical parameters and thus eliminating the

transmission of redundant data. In this chapter we present a distributed scheduling algorithm that decides when a particular node should perform this novel type of aggregation[1,2]. The scheduling algorithm autonomously reassigns schedules when changes in network topology due to failing or newly added nodes, are detected. Such changes in topology are detected using cross-layer information from the underlying MAC layer. We first present theoretical performance bounds of our algorithm. We then present simulation results which indicate a reduction in message transmissions of up to 85% and an increase in network lifetime of up to 92% when compared to collecting raw data. Our algorithm is also capable of completely eliminating dropped messages due to buffer overflow.

[1] 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. In *ACM Transactions on Sensor Networks*, To appear.

[2] S. Chatterjea, T. Nieberg, Y. Zhang and P. Havinga. Energy-Efficient Data Acquisition using a Distributed and Self-organizing Scheduling Algorithm for Wireless Sensor Networks. In *Proceedings of the Third IEEE Conference on Distributed Computing in Sensor Systems* (DCOSS), June 2007, Santa Fe, USA. pp. 368-385. Lecture Notes in Computer Science 4549 (LNCS4549). Springer Verlag.

5.1 Introduction

Wireless sensor networks (WSNs) are increasingly being used to carry out various forms of environmental monitoring. Monitoring vineyards [41], wildlife habitats [98], office buildings [142], suspension bridges [131], forests [132] and even marine environments [47] are just a few of the diverse range of sensor network applications that can be found in current literature. One of the primary motivations for using WSNs is that they allow environments to be monitored at extremely high spatial and temporal resolutions - something that is not possible using existing monitoring technologies. This is mainly due to the fact that sensor nodes are usually deployed in very high densities [80].

However, extracting the vast amounts of data generated by large-scale, high-density sensor network deployments can cause a wide range of problems. The fact that sensor nodes are typically battery powered devices makes energy resources a precious commodity. Transmitting every single acquired sensor reading would cause nodes to drain their batteries in a matter of days. WSN deployments, however, will only be practically viable if they are able to run unattended for long durations. Furthermore, the limited bandwidth of present-day sensor nodes prevents all the acquired readings from being propagated successfully towards the sink. This results in dropped packets, which in turn has a negative impact on the quality of data collected.

As sensor readings of adjacent nodes in a high-density network may display a high degree of correlation, one way to reduce the amount of data that needs to be transmitted would be to exploit the spatial correlation between adjacent nodes. Thus instead of having every node transmit its readings, we suggest a method that requires only a particular small subset of nodes in the network to transmit messages that represent all the remaining nodes at any point in time. We refer to nodes belonging to this subset as *correlating nodes*. Every correlating node initially transmits a message containing correlation information that indicates how the particular node's readings are correlated with its adjacent neighbors. Subsequently, the correlating node continues to transmit its own readings until a change in correlation is detected, in which case, the updated correlation information is transmitted to the sink node. The sink node uses the correlation information and combines it with the subsequent reading received from a correlating node to deduce the readings of the adjacent neighbors of the correlating node. As it would be pointless to have two adjacent nodes to act as correlating nodes simultaneously, in this chapter we present a completely distributed and self-organizing scheduling algorithm that decides when a particular node should act as a correlating node. Our contributions are stated as follows:

1. We present a completely distributed scheduling algorithm that enables every node to autonomously choose schedules based only on locally available information.

2. We prove our algorithm possesses self-stabilizing properties that allow it to recover within a finite time regardless of any disturbances in the network such as topology changes or communication errors. We present theoretical upper bounds for message transmissions and network stabilization times when topology changes occur.

3. We illustrate how our algorithm is able to adapt quickly to topology changes due to its close interaction with the underlying MAC layer. The algorithm also improves energy-efficiency by taking advantage of cross-layer information provided by the MAC.

4. We present performance estimates and theoretical upper bounds for the performance of our algorithm. We evaluate the algorithm by presenting simulation results which indicate a reduction in message transmissions of up to 85% and an increase in network lifetime of up to 92% when compared to collecting raw data. Our algorithm is also capable of completely eliminating dropped messages due to buffer overflow.

A list of assumptions based on the GBR application scenario mentioned earlier in Section 2.3.1 is described in the following section. It is important to note, however, that our work is not strictly tailored for the GBR. As mentioned later in Section 5.3, it can be used in a wide range of environmental monitoring scenarios where fine-grained spatio-temporal resolutions are required. We have simply chosen to use the GBR as a test bed to illustrate the feasibility of our solution. Section 5.3 provides the motivation and focus of this chapter. An overview of our approach is presented in Section 5.4. Section 5.5 provides background information about self-stabilization. The main scheduling algorithm is described in Section 5.6. We evaluate the performance of our approach in Section 5.8. Section 5.9 mentions the related work and finally the chapter is concluded in Section 5.10.

5.2 Assumptions

Based on the GBR application scenario, we have made a few assumptions about the data that will be collected as well as the network itself. Firstly, as there will

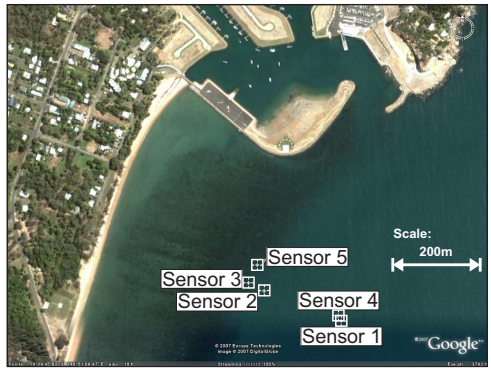

Figure 5.1: Sensors deployed in Nelly Bay, Great Barrier Reef, Australia

be a very large number of sensor nodes (~ 100) and since they may be required to obtain readings at a high frequency, a large amount of data can be expected to flow through the network. Given the limited bandwidth and memory capacity of individual sensor nodes, assuming that nodes are transmitting data via a communication tree towards the sink node, nodes that are closer to the sink node, will be prone to buffer overflows [64]. This will result in loss of messages and greatly reduce the quality of data collected. Secondly, as there will be a very high density of sensor nodes, i.e. they will be placed very close to each other, we can expect readings between neighboring nodes to be correlated during most parts of the day. This assumption can be verified by looking at data that has been collected from Nelly Bay in the GBR shown in Figure 5.1 [37]. Figure 5.2(a) presents a matrix that shows three characteristics of the 5 deployed sensors: temperature readings (d), correlation between the readings of any two sensors (c) and how correlation varies over time (b). It can be clearly seen that the correlation remains relatively constant over a 12 day duration. Note that temperature readings were obtained every 10 minutes.

As the sensor nodes will be placed on the reef for possibly a number of years, we assume that the topology of the network is relatively static. We do, however, take into consideration the fact that the network topology may change occasionally as the nodes are prone to failure (e.g. due to the harsh environment or dead batteries), and new nodes may be added to expand the network.

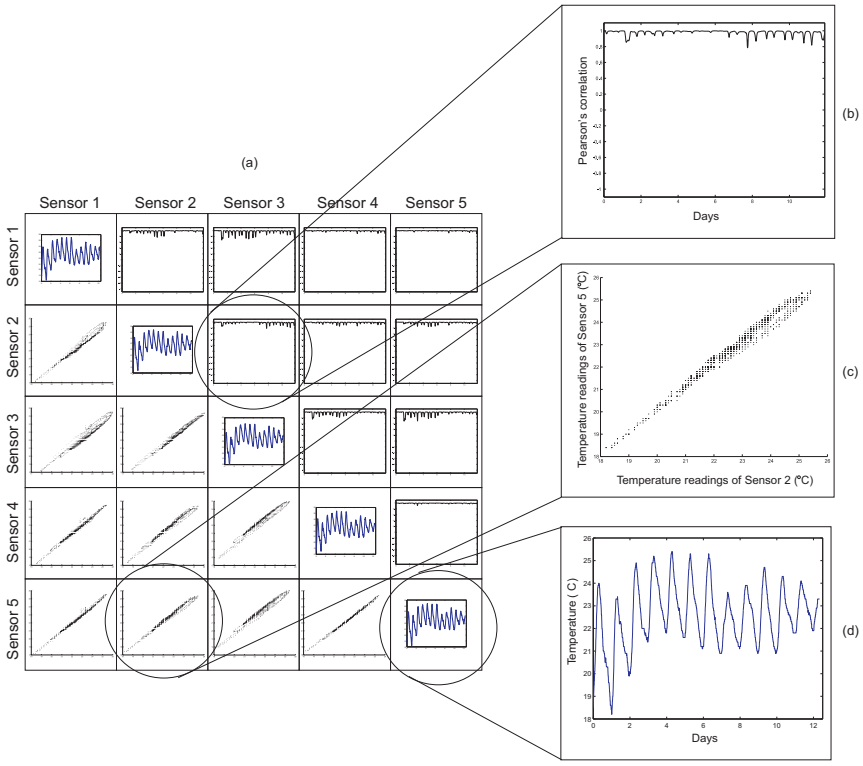

Figure 5.2: (a) Correlation matrix, (b) Variation of correlation over time, (c) Correlation between two sensor readings, (d) Temperature readings

5.3 Motivation and focus

Taking advantage of spatial correlations between neighboring nodes would enable nodes to filter out redundant data. This in turn will help reduce problems such as excessive energy usage, buffer overflows and reduced data quality. Instead of transmitting every acquired sensor reading to the sink node, a node which discovers a correlation with its neighboring nodes only transmits the correlation information followed by its own readings. Thus the sink node can then predict the readings of the neighboring nodes using the correlation information and the transmitted readings from the node performing the correlation. This is

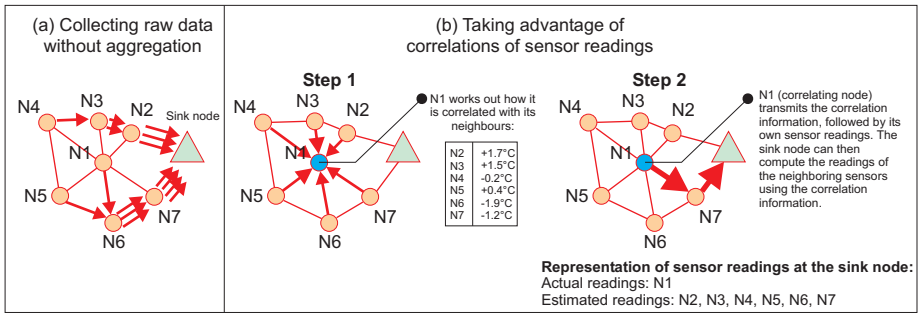

Figure 5.3: Advantage of using correlation information (b) instead of transmitting raw data (a)

illustrated in Figure 5.3(b).

The approach of taking advantage of spatial and temporal correlations of sensor readings involves two issues that need to be addressed:

- *Identifying correlations and keeping correlation information updated*: It is important to note that correlation is not a static attribute. Correlation between two neighboring sensors may exist at only certain times of the day. Thus a node needs to be able to identify when a correlation may arise and it also needs to ensure that the correlation information it has is up-to-date. Naturally, if trends of sensor readings change extremely rapidly, such a scheme would incur a very high overhead that would exceed the cost of collecting raw data from the network due to frequent updates of the correlation information. However, preliminary readings obtained from our four different sensor network test beds situated in diverse environments ranging from the coral reef, to microclimates in trees and even a typical office environment, have shown that sudden changes in trends of sensor readings are not particularly common. This characteristic is also clearly shown in Figure 5.2(b). In fact, during most parts of the day, sensors placed geographically close to one another, tend to display similar behavior. Our work is not designed for applications where correlations fluctuate rapidly.

- *Deciding when a node should act as a correlating node*: It would not make sense for all nodes to send correlation information to the sink node simultaneously as this would involve sending more information than even

transmitting raw sensor readings. Thus when one node is transmitting correlation data, the neighboring nodes should refrain from doing so. This implies that while nodes transmitting the correlation information (i.e. *correlating nodes*) are represented at the root node by their *actual* (own) readings, their neighbors however, are represented by *estimated* readings which are based on the correlation information transmitted by the correlating nodes (Figure 5.3(b)). Note that a correlating node initially transmits the correlation information followed by its own sensor readings. Thus two neighboring nodes should not act as correlating nodes simultaneously at any instant of time. Furthermore, it is important to ensure that at all times, every node in the network is represented at the sink node either by an actual reading or an estimated reading. This in turn means that if a node is not a correlating node at a certain time, it must be connected to at least one neighboring correlating node.

Having a static scheduling scheme which fixes the correlating nodes for the entire lifetime of the network is not desirable. This is because it would mean that while there are a number of correlating nodes sending their own sensor readings in addition to the correlation information, a significant proportion of the nodes would always be represented at the root node by only estimated readings. Thus such a scheme would be prone to errors in the event that the correlating node fails for some reason and starts sending erroneous correlation information to the sink.

Thus in order to have a more robust scheme, every node in the network should be given an opportunity to be a correlating node. This would allow the sink to raise an alarm in case it notices that the actual readings of a node display a distinctly different characteristic compared to the estimated readings of the same node.

This clearly implies that there needs to be a scheduling scheme which decides when a certain node should be in charge of sending correlation information in the event that a correlation exists.

The work in this chapter focuses on the *second* issue and presents a Distributed and self-Organizing Scheduling Algorithm ($\mathcal{DOSA}$) that allows nodes to autonomously re-assign the schedules if a change in topology is detected be it due to the failure or addition of nodes. We make the assumption in this chapter that correlations between neighboring sensor nodes do exist. The exact mechanisms for identifying correlations and keeping correlation models updated does not fall within the scope of this chapter.

5.4 A macro perspective of the $\mathcal{DOSA}$ approach

As we mentioned in Section 5.3, the primary objective of $\mathcal{DOSA}$ is to help decide *when* a particular node should act as a correlating node and thus be put in charge of representing the sensor readings of all the nodes in its one-hop neighborhood. Note that during the correlating node's schedule, the node initially transmits correlation information to the sink node followed by its own sensor readings. All the nodes in the correlating node's one-hop neighborhood, *do not* transmit their sensor readings to the sink during this period.

Since $\mathcal{DOSA}$ is intended to solve a scheduling problem, we make use of a distributed graph colouring algorithm to assign schedules to individual nodes [93]. Thus, from a graph theoretic point of view, since no two adjacent nodes can act as a correlating node simultaneously, all the nodes chosen by $\mathcal{DOSA}$ to be correlating nodes need to form an *independent set*. Additionally, the correlating nodes for a particular instant of time need to form a *dominating set* since every non-correlating node must be joined to at least one correlating node by some edge. Also note that the subset of nodes that is both independent and dominating is known as a *maximal independent set*. A maximal independent set cannot be extended further by the addition of any other nodes from the graph.

It is these requirements that help us define the constraints outlined later in Section 5.6 that $\mathcal{DOSA}$ follows in order to perform its intended task.

In order to hasten the speed at which the nodes are assigned schedules, $\mathcal{DOSA}$ makes use of the information provided by the underlying MAC protocol, LMAC [78]. In other words, instead of $\mathcal{DOSA}$ having to colour all the nodes from scratch, it takes advantage of the schedules (or colours) *already* assigned by LMAC and subsequently builds up on that to ensure that the requirements of $\mathcal{DOSA}$ are met. An added advantage of this form of cross-layer optimization is that a lesser number of messages need to be transmitted for all the schedules to be assigned properly as we make use of information that already exists. Furthermore, $\mathcal{DOSA}$'s dependence on LMAC makes it more reactive to changes in topology as any changes in neighborhood detected by LMAC are immediately filtered to $\mathcal{DOSA}$.

We refer the reader to Section 2.5.1 for a brief overview of the LMAC protocol.

5.5 Preliminaries for self-stabilization

As we later illustrate how $\mathcal{DOSA}$ initializes during start-up and how it is capable of recovering from topology changes due to the addition or removal of nodes, we take the self-stabilization [62, 63] approach to formalize the self-organizing properties of the algorithm. Self-stabilization allows a system that enters an illegitimate state (e.g. due to the occurrence of transient faults) to converge back to a legitimate state within a finite time without any external intervention. We now present some preliminaries of self-stabilization.

All nodes in the network are assumed to have unique IDs and have knowledge of their adjacent neighbors. Each node has a state that is specified by its local variables. The state of the entire system is called the *global state* or *configuration* and is the union of the local states of all the nodes. The objective of the system is to reach a desirable global final state called a *legitimate state*. The state of a system can either be *legitimate* or *illegitimate*. We use $\mathcal{S}$ to denote the set of all possible states. In order for the system to recover after a transient fault, all the affected nodes repeatedly execute a piece of code consisting of a finite set of rules having the form $(label)[guard] :< statement >;$. The statement part of the rule is the description of the algorithm used to compute the new values for local variables. A rule is *enabled* when its guard is true. The *execution* of an enabled rule determines the new state value of a node using the algorithm described by the statement part of the rule.

We denote the set of all legitimate states by $\mathcal{L}$ such that $\mathcal{L} \subseteq \mathcal{S}$. We denote the set of rules using $\mathcal{R}$ where $\mathcal{R} \in \mathcal{S} \times \mathcal{S}$ such that $(s_i, s_j) \in \mathcal{R}$. An execution of e is a maximal sequence of states, $e = s_i, s_{i+1}, ...s_j$ such that $\forall i \geq 1, s_i \in \mathcal{S}$, and s_i is reached from s_{i-1} by executing a particular rule.

A system can be considered to be self-stabilizing if the following two conditions hold:

- **Closure**: If $s \in \mathcal{L}$ and $s \rightarrow s'$ then $s' \in \mathcal{L}$. Therefore the closure property means that when a system is in a legitimate state, the following state is always a legitimate state as well, regardless of the rule executed.

- **Convergence**: Starting from any configuration $s \in \mathcal{S}$, every execution reaches $\mathcal{L}$ within a finite number of transitions.

The preliminaries presented above are used in the following sections to illustrate how $\mathcal{DOSA}$ is able to start-up properly and also how it is capable of recovering when the system experiences certain transient faults.

5.6 $\mathcal{DOSA}$: A distributed and self-organizing scheduling algorithm

$\mathcal{DOSA}$ uses a distributed graph colouring approach to decide when a particular node should be a correlating node. Every colour owned by a node represents a particular frame of time during which a node is required to act as a correlating node. In conventional graph colouring approaches, colours are assigned to vertices such that adjacent vertices are assigned different colours and the number of colours used is minimized. While $\mathcal{DOSA}$'s graph colouring approach also ensures that adjacent nodes in the network do not own the same colours it differs in the sense that each node is allowed to own *multiple* colours, i.e. a node can have multiple schedules. Moreover, the number of colours used in $\mathcal{DOSA}$ is fixed and is equal to the number of slots that are assigned to an LMAC frame.

Before we proceed, we first state certain definitions that are used through out the rest of this chapter.

We model the network topology as an undirected graph G where $G = (V, E)$. V represents the vertices or nodes in the network while two nodes are connected by an edge in E if they are within radio transmission range of each other. $\mathbf{K}$ represents the set of colours used to colour *all* the nodes. So $|\mathbf{K}|$ is equal to the number of slots per frame in LMAC. Also, we denote the *closed* neighbourhood of a node $v \in V$ by $\Gamma(v)$ i.e.

$$\Gamma(v) := \{u \in V | (u, v) \in E\} \cup \{v\}.$$

Using the graph-theoretic distance $d_G(u, v)$, that denotes the number of edges on a shortest path in G between vertices u and v, we can define the r^{th} neighbourhood of v as

$$\Gamma_r(v) := \{u \in V | d_G(u, v) \leq r\}.$$

Similarly, we define the *open* neighbourhood of a node v by $\Gamma'(v)$ where:

$$\Gamma'(v) := \{u \in V | (u, v) \in E\}.$$

Given that $\Gamma'^{(v)}$ denotes the *open* neighbourhood of node v, we refer to C_v as the set of colours owned by node v. Then for C_v it holds that,

$$0 < |C_v| < (|\mathbf{K}| - |\Gamma'(v)|).$$

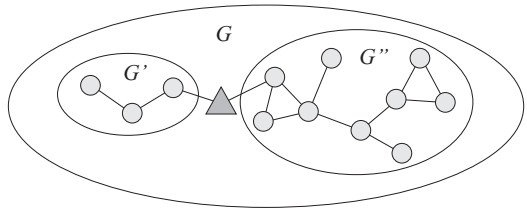

Figure 5.4: Two independent components in G

Given that a *node-induced subgraph* is a subset of the nodes of a graph G together with edges whose endpoints are both in this subset, we define a *component* as a node induced subgraph of a subset of nodes. Furthermore, we call two components independent if they are not connected by an edge. As an example, in Figure 5.4, G' and G'' are two independent components in G.

Before describing the details of the operation of $\mathcal{DOSA}$, we first state the constraints derived from the requirements stated in Section 5.4, which define its behavior. The following two constraints must be met when two nodes u and v are adjacent to each other:

Constraint 1: $C_v \cap C_u = \emptyset$

In other words, two adjacent nodes cannot own the same colours. This is because two adjacent nodes should not be assigned as correlating nodes in the same time instant.

Constraint 2: $C_{\Gamma(v)} = \mathbf{K}$

All colours should be present within the one-hop neighbourhood of node v, i.e. if node v does not own a particular colour itself, the colour ***must*** *be present in one of its neighbouring nodes that is one hop away. This ensures that every node's readings will be represented at the sink node for every time instant either directly or through a correlated reading.*

Lemma 5.1. *The combination of constraints 1 and 2 ensures that at any time slot, c_i, all nodes owning the colour c_i, which correspond to that time slot, form a maximal independent set on G.*

Proof. At any time instant according to Constraint 1, two adjacent nodes will never own the colour c_i, thus resulting in an independent set I. Constraint 2 ensures that in the closed neighbourhood of every node $v \in V$, every colour is present. This clearly results in a maximal independent set. □

5.6.1 Details of simulation setup

For the sake of easier comparison, the simulation results are presented immediately after the description of the theoretical performance bounds of $\mathcal{DOSA}$ in every subsection that follows. Thus we first state the salient details of our simulation setup and then proceed with the rest of the sections.

All simulations are implemented in Matlab [103]. Simulation results (unless otherwise specified) are averaged out over 100 randomly generated network topologies for a particular average node connectivity. Each topology consists of 100 nodes randomly distributed in a 100x100 unit area. The average connectivity (or neighbor density) has been varied from 5 to 11 by setting different transmission ranges for the nodes. Nodes are static and homogeneous in the sense that all the nodes have the same transmission radii. The number of slots per frame in the LMAC implementation is 32.

5.6.2 Dependency of $\mathcal{DOSA}$ on LMAC

As mentioned in Section 2.5.1, LMAC assigns a slot to every node in the network. $\mathcal{DOSA}$ begins its distributed colouring scheme by considering the initial slot assignment phase in LMAC as an input. Slot assignments in LMAC correspond to partial colour assignments in $\mathcal{DOSA}$. Thus while LMAC assigns every node with a single colour, $\mathcal{DOSA}$ assigns the remaining colours that ensure the adherence to the constraints 1 and 2 given in the previous section. We can then state that,

$$C_v = C_{v_{LMAC}} \cup C_{v_{DOSA}}$$

where $C_{v_{LMAC}}$ refers to the colour corresponding to the LMAC slot owned by node v and $C_{v_{DOSA}}$ refers to the colours assigned to node v by $\mathcal{DOSA}$.

Similarly, the colours owned by the nodes adjacent to node v, $C_{\Gamma'(v)}$, are also made up of LMAC and $\mathcal{DOSA}$ colours. Thus we can state,

$$C_{\Gamma'(v)} = C_{\Gamma'(v)_{LMAC}} \cup C_{\Gamma'(v)_{\mathcal{DOSA}}}.$$

The dependency of $\mathcal{DOSA}$ on LMAC allows nodes to adapt autonomously and immediately to changes in network topology. For example, the addition or removal of a node results in the change being reflected in the LMAC Neighbour Tables of all other neighbouring nodes within range. $\mathcal{DOSA}$ detects changes in LMAC's Neighbour Table and performs a re-assignment of schedules if any of the neighbouring nodes do not meet the constraints mentioned above. Utilizing

such cross-layer information from LMAC ensures that $\mathcal{DOSA}$ does not spend additional resources trying to detect topology changes itself.

We also make the assumption that the maximum degree (i.e. number of adjacent nodes) of a single node in the network is always known prior to deployment. This information is used to choose the appropriate number of slots in a particular frame in LMAC. In case the maximal degree of the nodes cannot be bounded accurately enough, LMAC also offers functionality to operate nodes passively, i.e. without owning a time-slot, when the network gets (locally) dense (see cf. [112]). However, for ease of notation and argumentation, we only consider active nodes that are assumed to acquire a free slot when carrying out slot assignment. The proper operation of LMAC also guarantees the proper operation of $\mathcal{DOSA}$.

5.6.3 General operation of $\mathcal{DOSA}$

$\mathcal{DOSA}$ uses a *greedy* approach to assign colours to nodes. Colouring is performed using two types of colours: *LMAC Colours* and $\mathcal{DOSA}$ *Colours*. LMAC Colours refer to the colours that have been assigned by LMAC - due to the slot assignment. $\mathcal{DOSA}$ Colours refer to the additional colours that are assigned by $\mathcal{DOSA}$ to ensure that constraints 1 and 2 are met. This occurs *after* the LMAC colours have been assigned. $\mathcal{DOSA}$ does not have any control over the LMAC Colour of a node as it depends purely on the slot assignment performed by LMAC. In fact, such control is also not required. Therefore, in the following, we refer to $\mathcal{DOSA}$ Colours simply as colours unless otherwise indicated.

Colours are acquired based on a calculated priority. A node computes its priority within its one-hop neighborhood based on its degree and node ID. The higher the degree of a node, the higher its priority. If two neighboring nodes have the same degree, priority is calculated based on the unique node ID; the node with the larger node ID will have the higher priority. This priority computation is performed in Line 4 of Algorithm 1.

Once all nodes have acquired their LMAC slots, a *BeginSecondPhase* message is injected into the network through the sink node requesting the nodes to begin the $\mathcal{DOSA}$ colouring phase. At this stage, every node receiving the *BeginSecondPhase* message only has an LMAC Colour and does not satisfy the constraints mentioned earlier. Thus these nodes mark themselves as `Unsatisfied`. A node only attains the `Satisfied` status when it satisfies the two constraints mentioned in Section 5.6. Upon receiving the *BeginSecondPhase* message, a node broadcasts the *NodeStatus* message. This message contains information about the node's degree, status (i.e. Satisfied/Unsatisfied) and the list of colours

Algorithm 1 $\mathcal{DOSA}$ - Normal Initialization

Input: NodeStatusMSG(Degree, SatisfiedStatus(TRUE/FALSE),ColoursOwned)
Output: NodeStatusMSG(Degree,SatisfiedStatus(TRUE),ColoursOwned)/NIL

```
 1: UPDATE(LocalInfoTable, v)
 2: if LocalInfoTable contains entries from ALL adjacent nodes then
 3:    if SatisfiedStatus(v)=FALSE then
 4:       Compute PRIORITY(v)
 5:       if PRIORITY(v)=Highest then
 6:          C_v ← K\C_Γ'(v)
 7:          ColoursOwned← C_v
 8:          SatisfiedStatus← TRUE
 9:          UPDATE(LocalInfoTable, v)
10:          BROADCAST NodeStatusMSG(Degree, SatisfiedStatus, ColoursOwned)
11:       end if
12:    end if
13: end if
```

owned. The *ColoursOwned* field is a string of $|\mathbf{K}|$ bits where every colour owned by a node is marked with a 1. The rest of the bits are marked with a 0. Initially, a node only marks its own LMAC Colour as 1 due to the initial LMAC slot assignment. A neighboring node that receives the *NodeStatus* message then performs colouring using $\mathcal{DOSA}$ as outlined in Algorithm 1. Note that the *NodeStatus* message is the only message that is used for the operation of $\mathcal{DOSA}$.

We now briefly describe the operation of $\mathcal{DOSA}$ outlined in Algorithm 1. Upon receiving a *NodeStatus* message, a node first updates its *LocalInfoTable* (Line 1). This table stores all the information contained in the *NodeStatus* messages that are received from all the adjacent nodes. Once a node has received *NodeStatus* messages from *all* its immediate neighbors (Line 2), and if its status is `Unsatisfied`(Line 3), the node proceeds to compute its priority. PRIORITY computes the priority of a node *only* among its unsatisfied neighbors (Line 4), i.e. as time progresses and more nodes attain the `Satisfied` status, PRIORITY needs to consider a smaller number of neighboring nodes. The highest priority is given to the node with the largest degree among its adjacent `Unsatisfied` neighbors. If more than one node has the same degree, then the highest priority is given to the `Unsatisfied` node with the largest NodeID.

The node that has the highest priority among all its immediate unsatisfied neighbors, acquires *all* the colours that are not owned by any of its adjacent neighbors (Line 7). As the node has then satisfied both constraints of $\mathcal{DOSA}$, it switches to the `Satisfied` state, updates its own *LocalInfoTable* and informs

all its neighbors through a broadcast operation (Lines 8-10). Note that this technique corresponds to a highest degree greedy approach.

Figure 5.5 provides a step-by-step example of how the $\mathcal{DOSA}$ algorithm assigns colours to the nodes in a network. We make the assumption in the example that LMAC uses 16 slots.

5.6.3.1 Correctness of $\mathcal{DOSA}$

In this section we illustrate how $\mathcal{DOSA}$ is able to successfully carry out initialization within a finite time given any arbitrary network. We initially assume that no transmission errors occur throughout the initialization phase but subsequently describe how such issues are handled in Section 5.6.3.2.

In order for $\mathcal{DOSA}$ to operate properly, it is absolutely imperative that every node always has up-to-date state information of its immediate neighbors. If a node n experiences a certain change in state (e.g. change from `Satisfied` to `Unsatisfied`) and fails to inform an adjacent neighbor of the change, this neighbor node might execute certain inappropriate steps based on its outdated state information of n. This error may prevent $\mathcal{DOSA}$ from stabilizing within a finite time. Thus it is essential for $\mathcal{DOSA}$ to possess the *cache coherence* property [76].

Let each node $v \in V$ in the sensor network have a variable, C_v indicating the colours owned by node v. For each $(u, v) \in E$, let u have a variable $\bowtie_u C_v$ which denotes a cached version of C_v. We can call a system *cache coherent* at significant points in time, t, when at time t, $\forall u, v : (u, v) \in E : \bowtie_u C_v = C_v$ [76]. This means that whenever v assigns a value to C_v, node v also broadcasts the new value to all its neighbors. The moment a node u receives an updated value of C_v, it instantaneously (and atomically) updates $\bowtie_u C_v$.

If we consider the operation of LMAC alone, the cache coherency property does not hold. Let us consider the case where two adjacent nodes v and u own the slots i and j respectively where $j > i$. Suppose v first broadcasts its updated state information to u during its own slot i. Now consider the case where the state of v changes in slot l where $i < l < j$. In this case, v will be unable to broadcast its newly updated status to u as the earliest time when it can transmit will be in slot $i+n$ where n is the number of slots in a single frame, i.e. v would have to wait one entire frame. This delay in transmission prevents the cache coherence property from existing. Nevertheless, for $\mathcal{DOSA}$ we have the following lemma:

Lemma 5.2. *Assuming no errors occur, nodes executing the $\mathcal{DOSA}$ algorithm on top of the LMAC protocol are all* cache coherent.

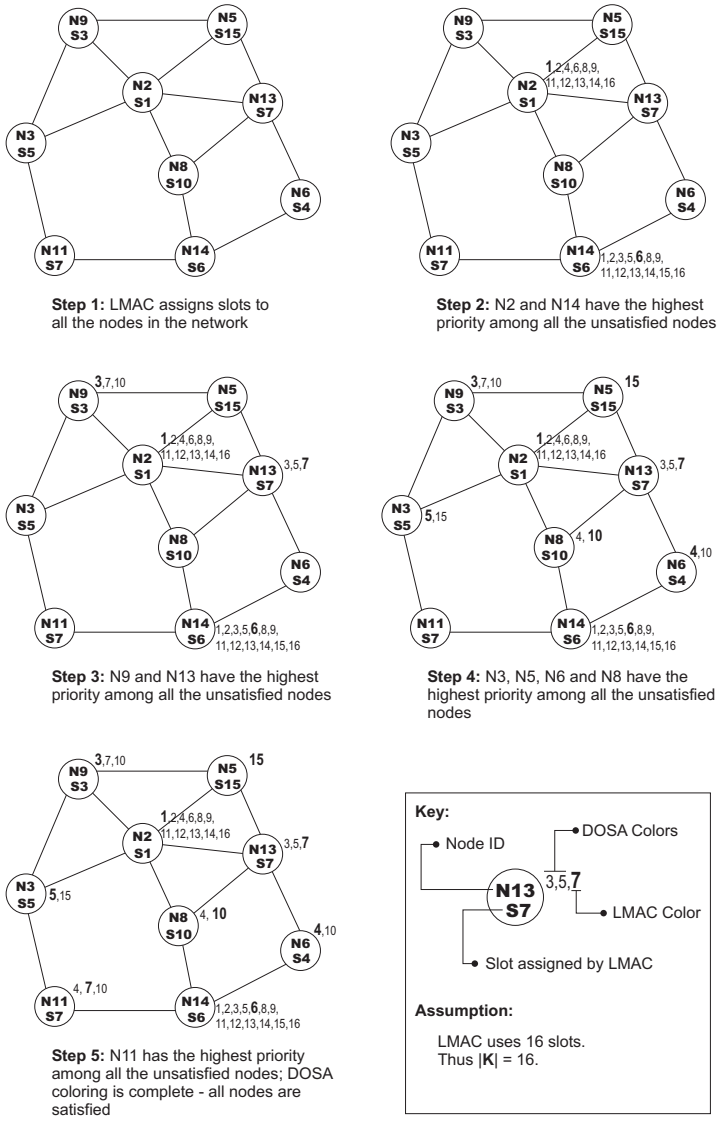

Figure 5.5: A step-by-step example of how $\mathcal{DOSA}$ colours are assigned

Proof sketch. In order to ensure cache coherence, $\mathcal{DOSA}$ carries out *pre-transmission state information processing* or PSIP. PSIP ensures that while a node updates its cache information the moment it receives updated state information from any adjacent neighbor, the node blocks any processing of the information in its cache until the point just before it transmits during its own slot. In other words, when a node receives updated state information from a neighboring node, it simply saves it. The node delays the processing of all the received state information till the point when the node is *just* about to transmit during its own slot. This effectively means that a node broadcasts any updated state change the moment it is detected and a node cannot experience a change in state at any time other than during its own slot. Thus while LMAC alone does not support cache coherence, PSIP guarantees that the state information used by $\mathcal{DOSA}$ is always cache coherent. □

There are a few properties that $\mathcal{DOSA}$ possesses that ensure that it stabilizes within a finite time: (i)Cache coherence (Shown in Lemma 5.2), (ii)Closure property, (iii)Convergence property. We describe the convergence and closure properties in greater detail below.

Lemma 5.3. *$\mathcal{DOSA}$ demonstrates both the closure and convergence properties.*

Proof. As we assume that no communication errors or topology changes occur during the initialization process, a node that acquires the `Satisfied` state, remains in that state forever, regardless of the messages received. This is synonymous to the closure property.

Recall from Section 5.5 that $\mathcal{S}$ denotes the set of all possible states. Let $\mathcal{M} \in \mathcal{S}$ (i.e. $\mathcal{S} \backslash \mathcal{M} = \mathcal{L}$) denote the set of all illegitimate states. In $\mathcal{DOSA}$, we consider all the nodes in the network that are *not* in the `Satisfied` state to belong to the set $\mathcal{M}$. Similarly, $\mathcal{L}$ represents all the nodes that have acquired the `Satisfied` state. $\mathcal{DOSA}$'s prioritization scheme, which is based on the combination of degree and ID of a node implies that a node can always compute a unique priority. This ensures that as long as $|\mathcal{M}| > 0$, in every atomic step, at least one node is enabled and thus attains the `Satisfied` state, i.e. if $n \in \mathcal{M}$, $|\mathcal{M}| = i$ and $|\mathcal{L}| = j$ in step r, then at step $r+1$, $n \in \mathcal{L}$, $|\mathcal{M}| = i - k$ and $|\mathcal{L}| = j + k$ where $k > 0$. Thus over a finite number of steps, all nodes in $\mathcal{M}$ eventually converge towards $\mathcal{L}$.

□

Lemma 5.4. *Assuming no transmission errors or topology changes occur, given that d is the number of nodes in G'_{max}, which is the largest independent compo-*

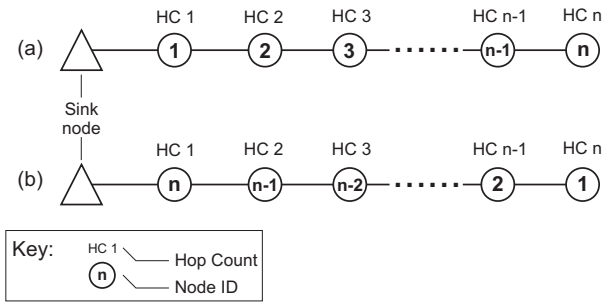

Figure 5.6: (a)Worst and (b)best case scenarios for $\mathcal{DOSA}$ initialization

nent in G, the time taken for all nodes in G to attain the `Satisfied` *state, t_s (in frames) in $\mathcal{DOSA}$ during the initialization is such that $d+1 \leq t_s \leq 2d-1$.*

Proof. As the $\mathcal{DOSA}$ initialization phase can run in parallel in separate independent components within a single graph G, and since the time taken for initialization to complete is dependent on the number of nodes, we can conclude that given a graph G, the initialization time is dependent on the cardinality of the largest independent component in G, i.e. G'_{max}.

From Figure 5.6(a) it can be seen that initialization takes the longest time when nodes in G'_{max} are arranged such that the smaller the hop count from the sink node, the smaller the node ID. In this example, node $n-1$ will have the highest priority and so all the nodes will reach the legitimate state only when node 1 receives the *NodeStatus* message from node 2. Given that there are d nodes in all, this occurs in frame $2d-1$ assuming that the sink node transmits the *BeginSecondPhase* message to node 1 in frame 1.

From Figure 5.6(b) it can be seen that initialization takes the shortest time when nodes in G'_{max} are arranged such that the larger the hop count from the sink node, the smaller the node ID. Thus a node at hop count d only acquires the `Satisfied` state when it receives the *NodeStatus* message from its adjacent neighbor at hop count $d+1$. This occurs in frame $d+1$. □

The node priorities which are computed locally by $\mathcal{DOSA}$ can be improved. Note that the greedy approach presented here works with any locally unique set of priorities. Especially in WSNs, where a sink node is present in the network, additional information based on distance to the sink can be exploited to obtain improved initialization times. For example, when two nodes have the same

degree, the one with the smallest hop count is given a higher priority. The idea behind this new prioritization is to ensure that nodes closer to the sink acquire the `Satisfied` status more quickly. In Figure 5.7, the effects of this optimized prioritization scheme are presented for different topologies based on simulations. These simulation results indicate that the average initialization times when varying the average connectivity from 5 to 11 can increase by up to 12%.

From the graphs in Figure 5.7 it can also be observed that $\mathcal{DOSA}$ initializes faster when the average connectivity (or network density) is smaller. Recall that in our simulations, we vary the average connectivity by increasing the transmission range of the nodes - *not* by increasing the number of nodes in the given area. Thus as the transmission range is increased, every node gets connected to a larger proportion of the nodes in the network. If we extrapolate this trend, i.e. set the transmission range to the length of the diagonal of the square area where the nodes are deployed, every node in the network can reach every other node within a single hop. In other words, this results in a complete graph. Note that in a complete graph, all the nodes would have the same degree, thus the prioritization scheme of $\mathcal{DOSA}$ would have to resort to using the NodeIDs to decide which node should be given the highest priority. This situation is similar to the scenario illustrated in Figure 5.6 since the entire process is serialized, i.e. a node with a smaller NodeID can only acquire colours once its adjacent neighbor with the next largest NodeID attains the `Satisfied` status. As we reduce the transmission range of the nodes, any single node would only be able to hear a subset of all the nodes in the entire network. In such instances, the DOSA initialization scheme may run in parallel in these separate subsets of nodes. This is the reason why the initialization of DOSA is faster when the average connectivity is smaller.

The graphs in Figure 5.7 also illustrate that the timing difference between the two prioritization schemes for a particular average connectivity reduces as the average connectivity reduces. As mentioned earlier, a reduction in the average connectivity means that the transmission range has been reduced. This in turn causes the diameter of the network in terms of hop counts to increase. Since there is a greater probability of having a small number of nodes with node IDs increasing in a linearly manner, and a smaller probability of having a large number of nodes with node IDs increasing linearly, the impact of the optimized prioritization scheme (i.e. including hop count when computing the priority) has a larger effect on the topologies with higher average connectivity.

Lemma 5.5. *During the initialization of DOSA, every node in the network*

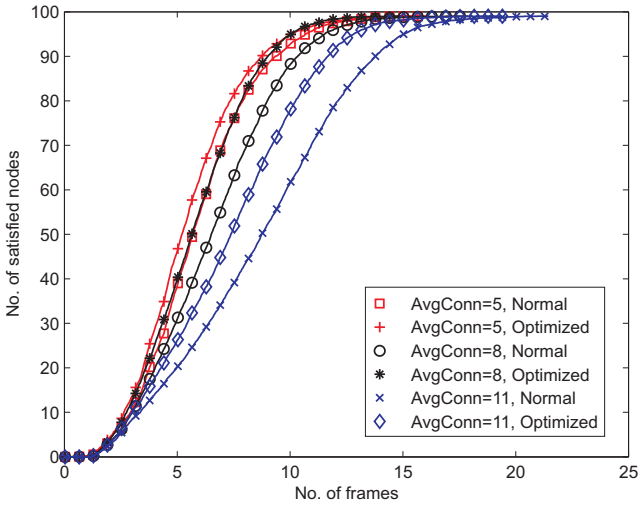

Figure 5.7: Difference in initialization times for the normal and optimized prioritization schemes (AvgConn: Average Connectivity, Normal: Highest priority given to largest degree, largest node ID, Optimized: Highest priority also considering smallest hop count)

transmits a total of 3 messages.

Proof. For the $\mathcal{DOSA}$ initialization to complete, every node in the network needs to broadcast a *BeginSecondPhase* message, a NodeStatus message with the SatisfiedStatus field set to $FALSE$ (broadcast when a *BeginSecondPhase* message is received) and finally a NodeStatus message with the SatisfiedStatus field set to $TRUE$ when a node attains the `Satisfied` state. Note that the number of messages transmitted by a single node is independent of the size of the network. □

5.6.3.2 Handling message corruption

Up to now, we have assumed that all communication is error free. However, to make our analysis realistic, we now describe the steps taken by $\mathcal{DOSA}$ to ensure that it continues to operate normally even when transmission errors due to poor link quality or topology changes do occur.

A node uses the Acknowledgement field in the CM section of a slot in LMAC to indicate whether it has successfully received an incoming message. Recall that since this field is in the CM section, every node transmits it once every frame. The number of bits in the Acknowledgement field corresponds to the total number of slots used in a frame. Thus if a node n receives a message successfully from a particular neighbor m in slot i, a '1' is placed in the i^{th} bit of the Acknowledgement field in the CM section. Similarly, a '0' is placed in the i^{th} bit if the incoming message received in slot i becomes corrupt. Node m can resend the message if it notices a '0' in the i^{th} bit of the Acknowledgement field of the CM received from node n.

Formally we state that every node n uses a boolean $b_n(m)$ for each neighbor m. For moving from a statement G to A in $\mathcal{DOSA}$ (Refer to Section 5.5 for definition of *statement.*), we can then state $(\forall m : (n, m) \in E : b_n(m)) \wedge G \rightarrow A$. If n receives a message correctly from a neighbor m, n assigns $b_n(m) := true$. If the message gets corrupted, $b_n(m) := false$ for every m. Thus n blocks the execution of $\mathcal{DOSA}$ the moment it receives a corrupt message and only continues executing the program once it has correctly received messages from all the neighbors.

Additionally, up to this point we have assumed that no topology changes occur during the initialization process. We would like to point out that this assumption was made simply to allow the initialization mechanism to be explained in a simpler manner. If a topology change does occur, e.g. a node disappears or reappears, $\mathcal{DOSA}$ makes use of the algorithms described in Sections 5.7.3 and 5.7.4 (which handle node removal and addition respectively) in order to ensure that the system continues to operate properly and eventually completes the initialization phase.

5.7 Performance of $\mathcal{DOSA}$

In this section we initially investigate the effectiveness of $\mathcal{DOSA}$ in several ways. First we observe the reduction in the number of nodes generating readings as compared to raw data collection. We also illustrate through simulations, how this reduction in message transmissions translates into longer network lifetime and also improved data quality.

Following this, we describe the behavior of $\mathcal{DOSA}$ when a node dies or is added to the network.

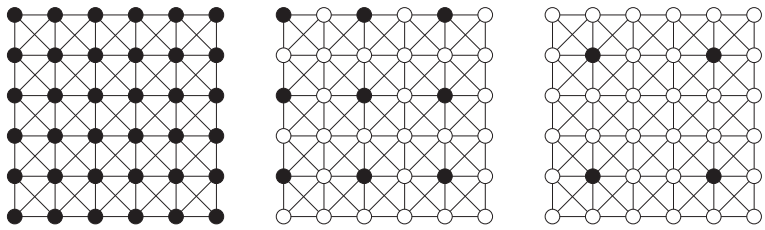

Figure 5.8: Impact of cardinality of maximal independent set

5.7.1 Effectiveness of $\mathcal{DOSA}$ in terms of message generation

The effectiveness of $\mathcal{DOSA}$ can be evaluated by observing the number of correlating nodes at any point of time and comparing it against the case of raw data collection where every node will be involved in transmitting raw sensor readings, Figure 5.8(a). Let us consider the two graphs in Figure 5.8(b) and (c). The black nodes, representing correlating nodes in both graphs form maximal independent sets. However, it can be seen that the cardinality of the maximal independent set can vary greatly depending on the set of chosen nodes. This results in varying degrees of energy efficiency since a larger cardinality means lower efficiency as compared to raw data collection.

This then leads us to the following question: *Given a particular graph, what is the maximum cardinality of the maximal independent set formed by $\mathcal{DOSA}$?* This would essentially give us an estimation or bound on the worst case performance of $\mathcal{DOSA}$. Since computing the maximum maximal independent set of a given graph is NP-hard [56], we take a "covering" approach to give a bound on the worst case performance of $\mathcal{DOSA}$.

Lemma 5.6. *The worst case performance of $\mathcal{DOSA}$ can be guaranteed to result in a message reduction of at least $(1 - \frac{xy}{2nr^2}) \times 100\%$ compared to raw data collection when n nodes are uniformly distributed in an area of dimensions $x \times y$ and every node has a circular transmission radius of r.*

Proof. Let us divide the area $x \times y$ into m squares where,

$$m = \frac{xy}{2r^2} \tag{5.1}$$

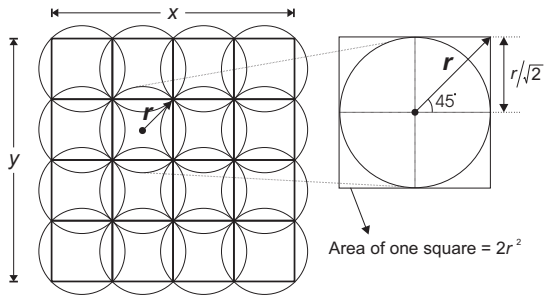

Figure 5.9: Estimating the cardinality of the maximum maximal independent set generated by $\mathcal{DOSA}$

Since the nodes are assumed to be randomly distributed, we may reasonably assume that nodes are present in all m squares, Figure 5.9. Note that this results in a worst-case estimation. Furthermore, we assume that exactly one node in every square forms part of a maximal independent set. We immediately see that it is not possible to have more than one node which is part of the maximal independent set in a single square as these 'extra' nodes would be in range of the first chosen node. Thus this consequently implies that the cardinality of the maximal independent set would be m. It would be impossible to increase the size any further by adding any more nodes. We can then conclude that the maximum cardinality of the maximal independent set created by $\mathcal{DOSA}$ is m. Thus the percentage in message reduction of $\mathcal{DOSA}$ compared to the collection of raw data would then be, $\frac{n-m}{n} \times 100$. This can then be simplified to $(1 - \frac{xy}{2nr^2}) \times 100\%$. □ □

As stated in [40], network density, μ can be defined as follows:

$$\mu = \frac{n\pi r^2}{xy} \tag{5.2}$$

Using equations 5.1 and 5.2, we can then state,

$$|I| \leq \frac{n\pi}{2\mu} \tag{5.3}$$

where I is any independent set also including the one computed by $\mathcal{DOSA}$. We would like to indicate however, that network density is approximately equal to average connectivity such that,

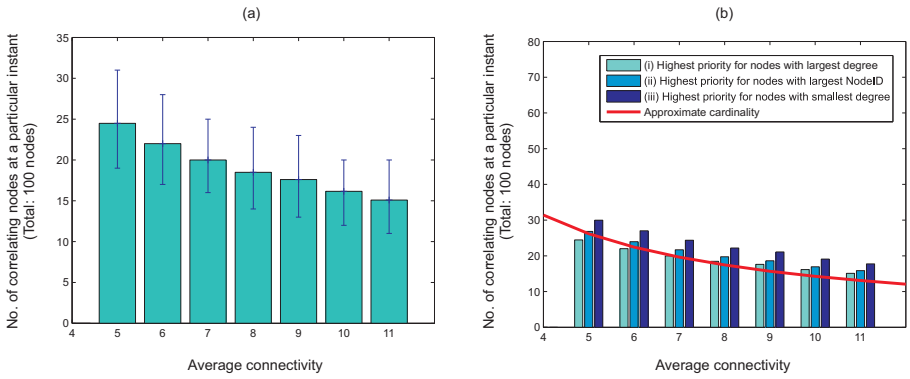

Figure 5.10: (a) Impact of average connectivity on the number of correlating nodes at a particular instant (Total number of nodes in the network = 100), (b) Effect of prioritization scheme on cardinality of maximum independent set

$$\frac{n\pi}{2\mu} \approx \frac{n\pi}{2(\rho+1)} \tag{5.4}$$

where ρ is the average connectivity. This result is used to plot the graph in Figure 5.10(b) which estimates the cardinality of $\mathcal{DOSA}$ as the average connectivity is varied.

The simulation results presented in Figure 5.10(a), show that even for a high cardinality, the number of correlating nodes is never greater than around 31% thus resulting in a reduction of message transmissions of around 69% compared to collecting raw data from every node in the network. This is of course for the cases where the average connectivity of the network is very low. As can be observed from Figure 5.10(a), the cardinality of the maximal independent set reduces further as the average connectivity of the network is increased. This is quite intuitive as a node can be used to represent a larger number of adjacent neighbors as the connectivity increases. The average reduction in message transmissions due to $\mathcal{DOSA}$ compared to raw data collection goes up to around 85% when the connectivity is increased to 11.

The prioritization scheme used in $\mathcal{DOSA}$ also has a large impact on the performance of the algorithm. We can observe two characteristics from the fact that $\mathcal{DOSA}$ gives the highest priority to the nodes with the largest connectivity. First, as nodes which have the highest degree in their local 1-hop neighborhood acquire the colours first following a greedy approach, the cardinality of the

maximal independent set tends towards the minimum maximal independent set. In Figure 5.10(b) we illustrate the effects of using 3 different priority schemes: (i) Highest priority given to the node with largest degree, (ii) Highest priority given to the node with largest node ID, (iii) Highest priority given to the node with smallest degree. By following the same argument as scheme (i), scheme (iii) results in a maximal independent set which has a cardinality that is closer to the cardinality of the maximum maximal independent set. Scheme (ii), however, due to its random nature, still results in a maximal independent set, but does not tend towards the minimum or maximum cardinality. Note that the difference between the estimated cardinality and the actual results can be attributed to boundary effects.

It is important to note, however, that while the minimum maximal independent set would result in an optimal solution (i.e. smallest number of correlating nodes), and thus appear to be the most efficient in terms of energy efficiency, it is not something that $\mathcal{DOSA}$ strives to attain. At this point, we would like to remark that computing an optimal, i.e. minimum cardinality maximal independent set is NP-hard [57]. Therefore given the scarce resource limitations of WSNs, we resort to the presented, faster approach. However, in [112] it is shown that for wireless communication networks the greedy strategy of $\mathcal{DOSA}$ results in a constant-factor approximation with respect to the cardinality of an optimal solution.

5.7.2 Effectiveness of $\mathcal{DOSA}$ in terms of network lifetime and data quality

As mentioned previously, the transmission of raw sensor readings has a detrimental impact on network lifetime and also on data quality. The reduction in message generation described in the previous sub-section naturally leads to improvements in both these factors.

In this subsection, we have carried out simulations to illustrate the benefits of $\mathcal{DOSA}$ in terms of network lifetime and data quality. Note that we define network lifetime as the total time taken before the death of the first node in the network. In the simulations, LMAC uses a frame length of 8 seconds. We use the following specifications based on the RFM TR1001 [27] transceiver: Transmit - 36mW, Receive - 11.4mW and Standby - 0.7μW to compute network lifetime. We also assume that correlations between sensor readings remain constant during this interval. All results have been collected over 10 minutes and have been averaged over 100 topologies where each topology consists of 100 nodes. Readings for the various graphs have been collected at the following

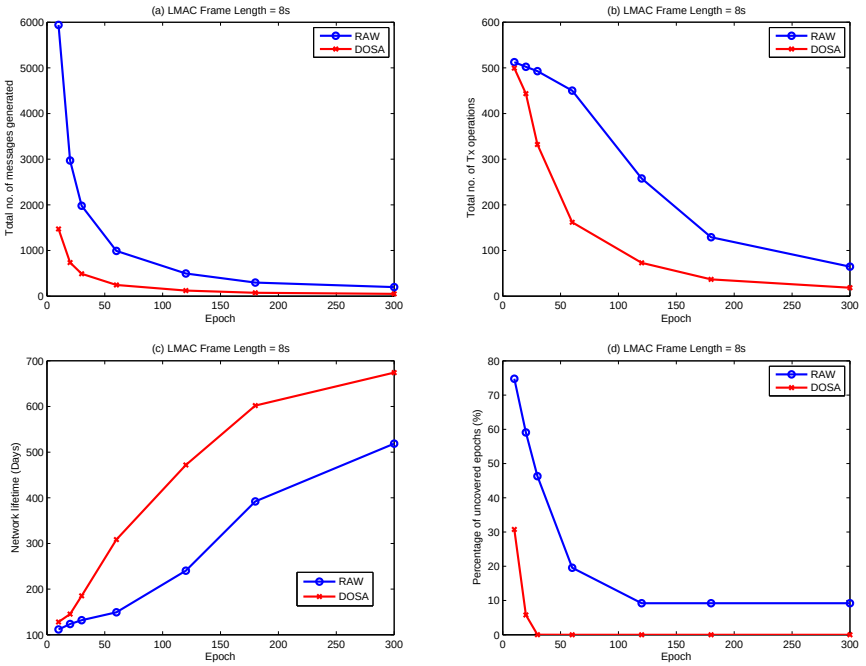

Figure 5.11: (a) Total number of messages generated, (b) Total number of transmit operations, (c) Network lifetime, (d) Percentage of uncovered epochs

epochs (in seconds): 10, 20, 30, 60, 120, 180, 300.

Figure 5.11(a) shows the total number of sensor readings that are generated during a 10 minute interval using both data collection techniques. Figure 5.11(b) shows the total number of transmit operations performed by all the nodes in the network for the entire duration of the simulation. One can clearly see that Figures 5.11(a) and 5.11(b) do not have similar shapes. This is primarily because both raw data collection and $\mathcal{DOSA}$ experience heavy message losses for high sampling rates. The left hand side of the graphs in Figure 5.11(b) tend towards each other as the limit of the maximum throughput of LMAC is nearly reached. It is this same characteristic that gives the shape of the network lifetime graph in Figure 5.11(c). Since the total number of message transmissions are nearly the same for both data collection methods at high sampling rates, the network lifetime is also quite similar. It can be seen from Figure 5.11(c) that $\mathcal{DOSA}$ can

help network lifetime improve by up to 83.5% (Epoch = 120s) as compared to raw data collection.

Apart from helping to improve network lifetime, $\mathcal{DOSA}$ also has a significant positive impact on the quality of data collected. When analyzing dropped messages for both data collection scenarios, it is important to realize that every message generated under the $\mathcal{DOSA}$ scheme carries a lot more weight than a single message in the raw data collection process. This is because a single sensor reading transmitted by a node n under the $\mathcal{DOSA}$ scheme represents not only the reading of n but also those of its adjacent neighbors. Due to this reason, we analyze data quality by observing the number of epochs which are *not represented* at the sink instead of simply counting the number of dropped messages. As an example, suppose a message generated by node n representing its own reading and its neighbors, q, r, and s for the epoch E is lost on the way to the sink due to a buffer overflow event. This would mean that during epoch E, the sink would not have any readings for nodes n,q, r, and s. Based on this example, we present the results of data quality in Figure 5.11(d). At high sampling rates e.g. when the Epoch is 10s, raw data collection results in around 75% of uncovered epochs while $\mathcal{DOSA}$ results in only 30% uncovered epochs. Uncovered epochs under $\mathcal{DOSA}$ quickly reduces to 0 and remains there as the sampling frequency is reduced. For raw data collection however, the percentage of uncovered epochs levels off at around 10%. We now explain this leveling off characteristic.

Usually a node drops messages when its buffers get filled up. Thus the higher the sampling rate, (i.e. the smaller the value of the Epoch) the larger the proportion of nodes in the network that experience buffer overflows. This naturally also increases the number of lost messages and in turn the percentage of uncovered epochs. However, as the sampling rate is reduced, the number of nodes experiencing buffer overflows may not keep on decreasing to zero. In most topologies, due to the simultaneous generation of messages by all nodes in the network, there will be a certain set of nodes that will *always* experience buffer overflows and will only allow a *fixed* number of messages to successfully traverse towards the root. Thus for low sampling rates, in every epoch, only a fixed number of messages will reach the root regardless of the chosen epoch. It is this characteristic that causes the percentage of uncovered epochs to level off for low sampling rates.

One may assume that the results from the graphs shown in Figures 5.11(a)-(d) clearly show that $\mathcal{DOSA}$ has a benefit only for applications that require low sampling rates. However, this is not the case. For applications that require high sampling rates and therefore high data rates, LMAC can easily be tuned such

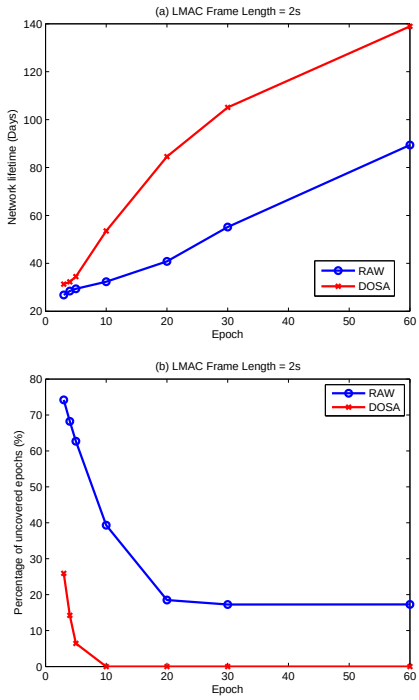

Figure 5.12: (a) Network lifetime, (b) Percentage of uncovered epochs

that one frame has a length of 2 seconds instead of 8 seconds. We illustrate the results of network lifetime and percentage of uncovered epochs in Figures 5.12(a) and (b). Note that these graphs also display the same characteristics mentioned above.

5.7.3 Coping with a dead node

As the death of a node could be a common occurrence in WSNs, it is important that any algorithm designed for WSNs is able to cope with such events. $\mathcal{DOSA}$ ensures that a node is able to reorganize the scheduling algorithm within a finite time autonomously the moment a neighboring node disappears from the network by retrieving cross-layer information from the underlying LMAC protocol, i.e. the death of a node triggers an update in the LMAC Neighbor Table.

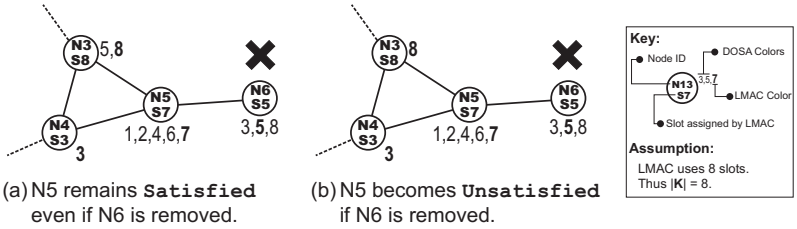

Figure 5.13: Two possible scenarios when a node dies

The death of a node leads to the disappearance of the colours that were owned by the dead node. This can lead to two possible scenarios. Firstly, it may be possible that one or more neighbors of the dead node still satisfy constraints 1 and 2 as the colours that have disappeared with the dead node are also present in its neighboring nodes. This is shown in Figure 5.13(a). In this case, the `Satisfied` neighboring nodes continue to maintain their existing schedules and do not transmit any messages. Note however, that while their colour assignments are invariant, the degree of the neighbors of the dead node does reduce by one. It is important that nodes that are one hop away from the neighbor of the dead node are informed about this change of degree as this information would be required in case any schedules need to be reassigned in the future due to certain network perturbations. However, as our design takes advantage of cross-layer information from LMAC, explicit message transmissions are not required to relay information regarding a change of degree of a node. This information is instead automatically disseminated through the periodic broadcast of the CM section of the LMAC protocol. Recall that the CM section transmitted by a node contains an Occupied Slot list which lists the slots occupied by the node and its one hop neighbors. Thus this information can also be used to deduce the degree of a node.

In the second scenario, shown in Figure 5.13(b), the death of a node may result in one or more neighboring nodes ending up with certain missing colours. As these nodes no longer satisfy constraints 1 and 2, the nodes switch to the `Unsatisfied` state and broadcast this change in status to their immediate one-hop neighborhood. A node then waits for one frame to see if there are any other neighboring nodes that are also in the `Unsatisfied` state. Note that waiting for one frame allows the node to hear from all its neighbors in case they have any status change to report. After waiting one frame, the node with the missing colour(s) acquires all the colours it lacks if it has the highest priority among all

Algorithm 2 $\mathcal{DOSA}$ - Coping with the loss of a node

Input: LMAC Neighbor Table indicates at least one missing node
Output: NodeStatusMSG(Degree, SatisfiedStatus(FALSE & TRUE), ColoursOwned)/NIL

```
1: UPDATE(LocalInfoTable, v)
2: if MissingColours(v) = TRUE (i.e. SatisfiedStatus(v)=FALSE) then
3:    BROADCAST NodeStatusMSG(Degree,SatisfiedStatus(FALSE), ColoursOwned)
4:    WAIT one frame
5:    Compute PRIORITY(v)
6:    if Priority(v)=Highest then
7:       C_v ← K\C_Γ'(v)
8:       ColoursOwned← C_v
9:       SatisfiedStatus← TRUE
10:      UPDATE(LocalInfoTable, v)
11:      BROADCAST NodeStatusMSG(Degree, SatisfiedStatus(TRUE), ColoursOwned)
12:   end if
13: end if
```

the unsatisfied nodes. This whole process is described in Algorithm 2. If a node lacks a colour but does not have the highest priority, it continues to wait until all its higher priority unsatisfied neighbors have become satisfied. In other words the node continues to execute Algorithm 1 every time it receives a *NodeStatus* message until it finally acquires the `Satisfied` state.

In order to explain the timing bounds of $\mathcal{DOSA}$ when a node dies, we use the same argument as in the proof of Lemma 5.4. We can extend this lemma as follows:

Lemma 5.7. *When a node v with x neighbors dies, the maximum time taken for all nodes to converge towards the* `Satisfied` *state is $x + 1$ frames where $x \leq |\boldsymbol{K}| - 1$.*

Proof. In the worst case, all the nodes of a dead neighbor switch to the `Unsatisfied` status and broadcast this change of state. Every `Unsatisfied` neighbor then waits for its higher priority `Unsatisfied` neighbor to switch to the `Satisfied` state before acquiring the `Satisfied` state itself. This situation is then identical to situation mentioned in Lemma 5.4 and thus the same timing bounds apply. □

We carried out simulations to compare typical network stabilization times when a node is removed with the bounds presented above. For every topology with 100 nodes (including one sink node), we first removed one node, waited for the network to stabilize (i.e. for all nodes to reacquire the Satisfied state) and then added it back to the network. This operation was carried out for all the

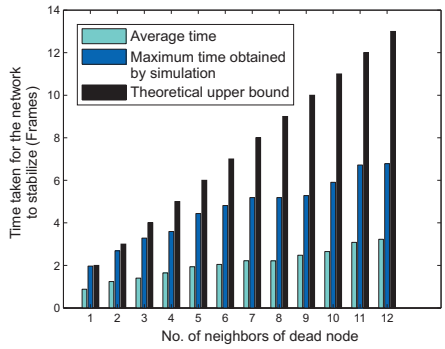

Figure 5.14: Time taken for a network to stabilize once a node has been removed from the network

99 nodes in every topology. Thus there were 9900 node removal and addition cycles. The results presented in the following sections have been obtained over these 9900 cycles. Note that the average connectivity of the nodes in every topology is 8.

Figure 5.14 presents the times taken for the network to stabilize once a node was removed from the network. Generally, the average stabilization time increases with the number of neighbors of the dead node. This is also true for both the maximum stabilization times and the theoretical upper bound presented above. However, as the number of neighbors of the dead node increases, the rate of increase of the average and maximum times decreases. This is because the probability of having a large number of nodes arranged in an increasing manner (e.g. Figure 5.6(b)) reduces as the number of neighbors increases. Thus in real life settings, a higher density network may not necessarily mean that the network would recover more slowly when a node is removed. In fact, according to the simulation results, the worst case recorded during simulation when the dead node has 12 neighbors is around 50% of the theoretical upper bound.

Lemma 5.8. *When a node v with x neighbors dies, the maximum possible number of messages that may be transmitted is $2x$ where $x \leq |\boldsymbol{K}| - 1$.*

Proof. As stated in Lemma 5.7 in the worst case, all x neighbors may become `Unsatisfied` when node v dies. Generally, every affected node (i.e. node with missing colours) initially transmits one *NodeStatus* message with the status set to `Unsatisfied` the moment node v dies. Finally, when a node acquires the Satisfied state, it transmits another *NodeStatus* message reflecting this change.

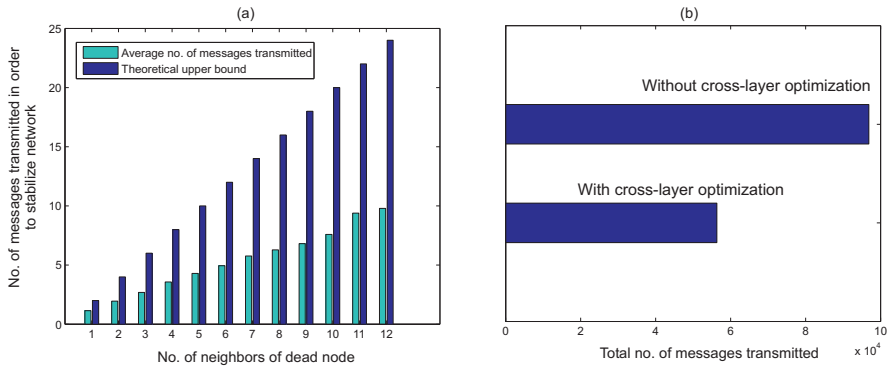

Figure 5.15: (a) Number of messages transmitted in order to stabilize the network once a node dies (b) Number of messages transmitted over 9900 runs with and without cross-layer information

Note that once a particular node acquires the Satisfied state, it remains there indefinitely. Thus the maximum possible number of messages that may be transmitted is $2x$. □

Figure 5.15(a) shows the average number of messages transmitted when a node with a particular number of neighbors is killed. Note that if all the neighbors become `Unsatisfied` due to the death of the node, every single neighbor will need to transmit 2 messages as explained earlier. In random network topologies however, the average number of messages transmitted when a node dies is less than 50% of the maximum theoretical upper bound indicated in Lemma 5.7.3.

The simulation results presented in Figure 5.15(b) show the benefit of having $\mathcal{DOSA}$ use underlying cross-layer information from LMAC. The total number of messages transmitted by all the nodes was compared over 9900 node deletions with and without cross-layer information being used. When it is not used, every neighbor of the dead node has to transmit a *NodeStatus* message regardless of its status. The results indicate a savings of up to 42% when cross layer information is used.

Lemma 5.9. *When a node v dies, only its first order neighbors may be affected, i.e. may switch from the* `Satisfied` *to the* `Unsatisfied` *state.*

Proof. The death of a node v can only result in the adjacent nodes experi-

encing missing colours and subsequently switching to the `Unsatisfied` state. `Unsatisfied` nodes then occupy colours they are lacking and thus ensure that their choice of colours will not cause any colour collisions with their neighbors. Also, any node that is `Satisfied` and receives a *NodeStatus* message, does not switch its status as long as Constraints 1 and 2 are met. Thus nodes that are 2 and more hops away from node v cannot experience a change of state when node v dies. □

5.7.4 Coping with a new node

As we illustrated in the previous sub-section, when a node dies, $\mathcal{DOSA}$ can only execute one fixed set of steps to ensure that the scheduling scheme stabilizes within a finite time. The node addition operation, however, is a little more involved as the set of steps taken by $\mathcal{DOSA}$ depends on the events that occur when a new node v is added to the network. For example, the node v may detect an LMAC collision or may cause colliding or missing colours in neighboring nodes or may even cause a combination of these events. Different permutations and combinations of these events can cause the network to react in a multitude of ways. This makes it impractical to analyze the performance bounds of every particular sequence of events that causes the network to react in a certain manner. Instead, in order to simplify matters, we categorize all the permutations and combinations of events depending on how far the network disturbance propagates when a node v is added to the network. For example, there may be a certain combination of events that would cause nodes that are up to 2 hops away from v to switch to the `Unsatisfied` state. Similarly there might be other events that would cause the network disturbance to propagate to the 3rd order neighborhood of node v.

We first begin by listing and describing the various events that could occur when node v is added. We have included an example in Figure 5.16 to illustrate how the various events might occur. (Note: The terms "LMAC slot" and "LMAC colour" are equivalent and thus can be used interchangeably.):

1. *Collision between LMAC slots*: This occurs when the new node v detects a collision between two (or more) of its adjacent neighbors. Every colliding neighbor then needs to give up the colliding slot and choose a new slot.

2. *Collision between LMAC colour (slot) and $\mathcal{DOSA}$ colours*: When a node n that is d hops away from node v, chooses a new LMAC colour, it causes a collision at an adjacent node m that is $d+1$ hops away from node v if

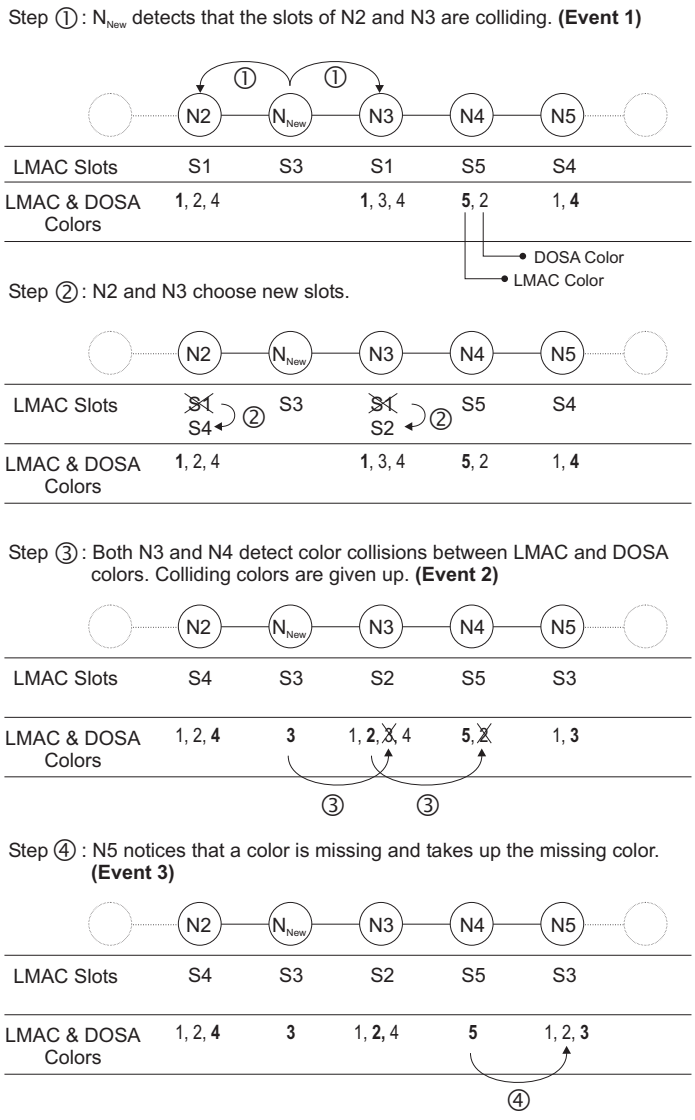

Figure 5.16: An example of how certain events occur when a new node is added.

node m owns the $\mathcal{DOSA}$ colour that is equal to the new LMAC colour chosen by node n. Note that if $d = 0$ then $n = v$. Also $0 \leq d \leq 1$ since using LMAC, node v can only detect slot collisions among its first order neighbors.

3. *Missing $\mathcal{DOSA}$ colours*: A node m that is $d + 1$ hops away from the new node v experiences missing $\mathcal{DOSA}$ colours if an adjacent node, n that is d hops away from node v gives up a $\mathcal{DOSA}$ colour due to a colour collision. Thus a colour collision at a node d hops from v can only cause missing colours at adjacent neighbors which are $d + 1$ hops away. Since a missing colour event at a node $d + 1$ hops away from v can only happen in combination with a colour collision event at an adjacent node d hops away from v, and since a colour collision can only occur in the 1st order neighborhood of v, we can conclude that $d \geq 1$ for a missing colour event to occur.

4. *Node obtains Highest priority (due to largest degree in local neighborhood)*: A node, n that is d hops away from the new node, v realizes that it has the highest degree in its local neighborhood after the addition of node v. This causes node n to obtain the Highest priority and thus acquire all colours except the LMAC colours of its adjacent neighbors. As this event can occur either at the new node itself (i.e. $d = 0$ and $n = v$) or at a node that is adjacent to v we can conclude that $0 \leq d \leq 1$.

However, the addition of a node does not cause a *domino* effect in $\mathcal{DOSA}$. The reason for this is explained in the following lemma:

Lemma 5.10. *When a node v is added, all nodes beyond the 3rd order neighborhood of v can be **guaranteed** to be unaffected, i.e. nodes which are more than 3 hops away will always remain in the* `Satisfied` *state regardless of the sequence of events that occur after the addition of node v.*

Proof. We know from the four events listed above that a node can switch to the `Unsatisfied` state when it experiences either a colour collision or a missing colour event. As explained above, a missing colour can only occur one hop away from a colour collision. We also know that a colour collision can occur up to a maximum of two hops away from v. This implies that a missing colour event can only happen in a node that is three hops away from v. Thus nodes more than 3 hops away from v cannot be affected by its addition. □

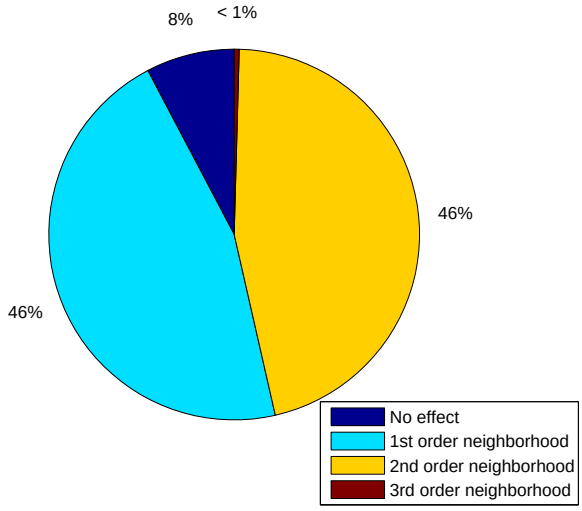

Figure 5.17: Simulation results showing how often a newly added node affects neighboring nodes 1 to 3 hops away from the new node.

While Lemma 5.10 shows that a node addition cannot cause $\mathcal{DOSA}$ schedules to be disturbed more than three hops away from the newly added node, we also carry out simulations to analyze the actual effects of node addition.

We perform simulations over 100 topologies each consisting of 100 randomly placed nodes. For every topology, we add a node randomly to the network and collect the required statistics, e.g. network stabilization time, depth of network disturbance, etc. This procedure is carried out for 100 nodes per topology. Thus the following results presented have been averaged out over 10,000 node additions.

Our simulations results presented in Figure 5.17 indicate that in around 92% of the cases, the network disturbance is restricted to *within* the second order neighborhood of the newly added node. In 8% of the cases, none of the neighbors are affected. Third order neighbors are only affected in less than 1% of the cases.

Regardless of which sequence of events occurs once a new node joins the network, initially, there are a few common steps that $\mathcal{DOSA}$ takes. Once these common steps are complete, the next set of steps taken depends on how far the

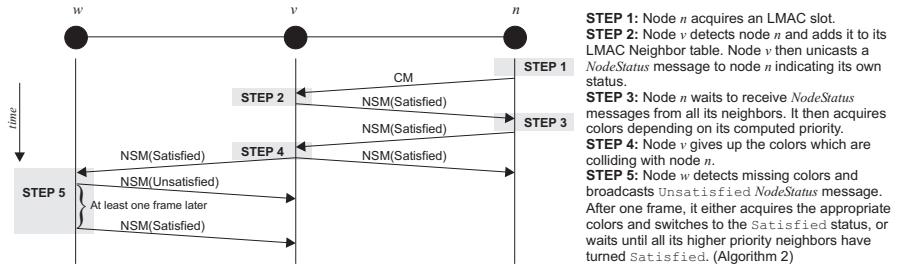

Figure 5.18: Timing diagram for addition of a new node, n (Node v is adjacent to n and node w is 2 hops from n. Note that NSM stands for *NodeStatus* message.)

Algorithm 3 $\mathcal{DOSA}$ - Coping with a new node

Input: NodeStatusMSG(Degree,SatisfiedStatus(TRUE),ColoursOwned)
Output: NodeStatusMSG(Degree,SatisfiedStatus(TRUE),ColoursOwned)

```
1:  UPDATE(LocalInfoTable,n)
2:  if LocalInfoTable contains entries from ALL adjacent nodes then
3:      Compute PRIORITY(n)
4:      if PRIORITY(n)=Highest then
5:          C_n ← K\C_{Γ'_LMAC(n)}
6:      else
7:          C_n ← K\C_{Γ'(n)}
8:      end if
9:      UPDATE(LocalInfoTable,n)
10:     BROADCAST NodeStatusMSG(Degree, SatisfiedStatus(TRUE), ColoursOwned)
11: end if
```

network disturbance will propagate. We first explain the initial common steps below.

When a new node, n is added to the network, LMAC ensures that the node n occupies a slot that is not used by any other node that is within 2 hops of n, Figure 5.18, Step 1. Node n then begins broadcasting its CM section. Neighboring nodes then detect the new node and add its entry into their LMAC Neighbor Table, Figure 5.18, Step 2. We explain the remaining steps taken by $\mathcal{DOSA}$ by referring to a neighboring node of node n as node v. It is also explained in Algorithm 3.

The moment v, which is already in the `Satisfied` state, detects a new node, n, it unicasts a *NodeStatus* message to node n, Figure 5.18, Step 2. Node n

Algorithm 4 $\mathcal{DOSA}$ - Colliding colours due to a new node

```
1: UPDATE(LocalInfoTable,n)
2: if LocalInfoTable contains entries from ALL adjacent nodes then
3:    if C_n ∩ C_v ≠ φ then
4:       C_v ← C_v\(C_n ∩ C_v)
5:       UPDATE(LocalInfoTable, v)
6:       BROADCAST NodeStatusMSG(Degree, SatisfiedStatus(TRUE), ColoursOwned)
7:    end if
8: end if
```

then waits to receive Node Status messages from all its adjacent neighbors, Figure 5.18, Step 3. Note that by this stage, n would know about the existence of all its adjacent neighbors as otherwise, it would not have been able to obtain an LMAC slot.

From this point onwards, the actions taken by $\mathcal{DOSA}$ are dependent on the sequence of events that occur. Once n has received *NodeStatus* messages from all its adjacent neighbors, it checks if it has the highest priority within its immediate neighborhood. If n finds that it has the highest priority, it acquires *all* colours *except* the LMAC colours of the adjacent neighboring nodes. This helps to ensure that over time, even if the network topology changes, the cardinality of the maximal independent set continues to be low. In other words, the sink node would be able to predict the readings of a larger number of nodes when a node with a higher degree is chosen to perform the correlations rather than a node with a very small degree. This leads to greater energy savings.

If however, node n realizes that it does not have the highest priority, it simply acquires all the colours that it is presently lacking. As node n has now satisfied constraints 1 and 2, it broadcasts a *NodeStatus* indicating that it is `Satisfied`.

At this stage, a neighboring node v, that receives the *NodeStatus* message from n may detect that certain colours are colliding, Algorithm 4, Line 3. This would mean that Constraint 1 is not being met. Thus node v gives up the colours that are colliding with node n, attains the `Satisfied` state, updates its own *LocalInfoTable* and informs all its neighbors through a broadcast operation, Figure 5.18, Step 4.

As v has had a change in colours, it could be possible that a node w, that is adjacent to v but not to n (i.e. w is 2 hops away from n), may become `Unsatisfied` (due to the Node Status message transmitted in Step 4 of Figure 5.18). Node w can then resolve the situation by executing Algorithm 2 which allows it to recover when certain colours are found to be missing, Figure 5.18,

	No effect (Group 1, 8%)	1st order (Group 2, 46%)
Event type	-	Colour collision
Max. time (Frames) Max Msgs Tx	≤ 1 $= \lvert\Gamma'_1(v)\rvert + 1$	≤ 1 $\leq 2\lvert\Gamma'_1(v)\rvert + 1$
	2nd order	
Event type	Colour collision (Group 3, <1%)	Missing colour (Group 4, 46%)
Max. time (Frames) Max Msgs Tx	≤ 2 $\leq 2\lvert\Gamma'_1(v)\rvert + 1 + \lvert\Gamma'_2 \setminus \Gamma'_1\rvert$	$2 + \lvert\Gamma'_2 \setminus \Gamma'_1\rvert$ $\leq 2\lvert\Gamma'_1(v)\rvert + 1 + 2\lvert\Gamma'_2 \setminus \Gamma'_1\rvert$
	3rd order	
Event type	Missing colour (Group 5, <1%)	
Max. time (Frames) Max Msgs Tx	$3 + \lvert\Gamma'_3 \setminus \Gamma'_2\rvert$ $\leq 2\lvert\Gamma'_1(v)\rvert + 1 + 2\lvert\Gamma'_2 \setminus \Gamma'_1\rvert + 2\lvert\Gamma'_3 \setminus \Gamma'_2\rvert$	

Table 5.1: Upper bounds for time and message transmission when a node is added

Step 5.

Next we present the upper bounds of $\mathcal{DOSA}$ in terms of time taken to stabilize the network and number of message transmissions when a new node is added. Since the addition of a node can result in the occurrence of several events, we breakdown the analysis into 5 possible groups, based on the depth of propagation of the network disturbance as shown in Table 5.1. Note that these 5 groups encompass all the possible sequence of events that can happen due to the addition of a new node, e.g. colliding LMAC slots, new node acquiring the highest priority, etc. Thus for example, the "3rd order, colour collision" event is not listed in Table 5.1 as such an event cannot happen due to the reasons stated in the list of events presented earlier in this section.

We refrain from explaining the derivations for the theoretical upper bound times for network stabilization shown in Table 5.1 as they have been derived using the same arguments presented earlier in Lemma 5.4. However, in order to present a more concise explanation, we present the theoretical upper bounds for the number of message transmissions using a set of 5 rules which are listed below:

- *Rule 1*: When a node that has already acquired $\mathcal{DOSA}$ colours detects a new neighbour node, v, it unicasts one *NodeStatus* message to node v.
- *Rule 2*: A new node, v broadcasts one *NodeStatus* message once it has acquired its LMAC slot, resolved all LMAC collisions amongst its neighbours

and has received *NodeStatus* messages from all its neighbours.

- *Rule 3*: A node, that acquires a new LMAC colour that is not listed in its existing list of DOSA colours broadcasts one *NodeStatus* message.
- *Rule 4*: A node that experiences a colour collision event transmits one *NodeStatus* message.
- *Rule 5*: A node that experiences a missing colour event transmits two *NodeStatus* messages - the first to indicate that the node is Unsatisfied due to the missing colour(s) and the second to indicate the node is Satisfied after it has acquired the appropriate colours.

We now illustrate with an example how the rules above can be used to work out the upper bound for the number of message transmissions for the "3rd order, missing colour" case:

- *Step 1*: First order neighbours of new node, v transmit a *NodeStatus* message after detecting it. ($|\Gamma'_1(v)|$ messages, *Rule* 1)
- *Step 2*: New node transmits *NodeStatus* message after acquiring colours. (1 message, *Rule* 2)
- *Step 3*: Every first order neighbour acquires new colours and broadcasts one *NodeStatus* message. ($|\Gamma'_1(v)|$ messages, *Rule* 3)
- *Step 4*: Every second order neighbour experiences a colour collision event and broadcasts one *NodeStatus* message. ($|\Gamma'_2(v) \setminus \Gamma\prime_1(v)|$ messages, *Rule* 4)
- *Step 5*: Every third order neighbour experiences a missing colour event and broadcasts two *NodeStatus* messages. ($2(|\Gamma'_3(v) \setminus \Gamma\prime_2(v)|)$ messages, *Rule* 5)

Thus,

Upper bound for total number of message transmissions for the "3rd order, missing colour" case $= 2(|\Gamma'_1(v)|) + 1 + |\Gamma'_2(v) \setminus \Gamma'_1(v)| + 2(|\Gamma'_3(v) \setminus \Gamma'_2(v)|)$.

Our simulations indicate that Groups 2 and 4 of Table 5.1 occur in 92% of all the 10,000 simulation runs while Group 1 occurs in 8% of the cases. Groups 3 and 5 however, occur in less than 1% of the cases. Thus we present the

simulation results only for Groups 2 and 4 since they form the most significant proportion of the various events that may occur.

We first consider the first order colour collision results. Figure 5.19(a) shows that regardless of the number of 1^{st} order neighbors a new node has, the network stabilization time remains within 1 frame. This coincides with the bounds stated in Table 5.1. Figure 5.19(b) shows that only around 1% of all the 1^{st} order colour collision cases resulted in scenarios where the number of messages transmitted was around 90-100% of the upper bound for message transmissions when a new node is added. In nearly 50% of the cases the number of messages transmitted was around 60% of the upper bound.

Next, we consider the second order missing colour results. Figure 5.19(c) shows that around 92% of time, the time taken for network stabilization when the second order nodes experience a missing colour event, was less than 40% of the upper bound. Figure 5.19(d) shows that in nearly 90% of cases, the number of messages transmitted was less than 60% of the upper bound. Notice that the results in Figure 5.19(b) tend closer to the upper bound than those presented in Figure 5.19(d). This is because while the results in Figure 5.19(b) only require the first order nodes to be affected, the results in Figure 5.19(d) involve both the first and second order nodes. Naturally the probability of affecting nodes both in the first and second order nodes is lower than affecting nodes in only the first order.

The overall performance of $\mathcal{DOSA}$ for node addition is presented in Figures 5.19(e) and 5.19(f). Figure 5.19(e) and Figure 5.19(f) show the distributions of the number of messages transmitted and the time taken for $\mathcal{DOSA}$ to stabilize once a new node is added to the network. It can be seen that in the majority of the cases, the network stabilizes within four frames.

5.8 Implementation of $\mathcal{DOSA}$

We evaluate the performance of $\mathcal{DOSA}$ by implementing the algorithm on 25 Ambient 2.0 μNodes [34]. While the footprint of LMAC and the AmbientRT operating system [34] is 2782 bytes, $\mathcal{DOSA}$ only takes up 869 bytes. All 25 nodes including the sink are within transmission range of each other, i.e. they form a complete graph. However, testing $\mathcal{DOSA}$ ideally requires a multihop network. As it is impractical to test a large number of different multihop topologies, we generate the topologies randomly and broadcast this information from the sink to the entire network. On receiving this information, each node uses it to create a virtual set of neighbors which is obviously a subset of the actual neighbors.

This results in a multihop topology. While LMAC uses the actual neighbors to perform slot allocation, $\mathcal{DOSA}$ uses the virtual set of neighbors to operate. We generate topologies for 5 different average connectivities ranging from 5 to 9. For each average connectivity, we create 50 random topologies and each topology consists of 25 nodes. Thus we have 250 topologies in total.

The implementation was used to carry out exactly the same experiments that have been described so far in the previous sections. However, as the results obtained are nearly identical to the simulation results, we have not included the results in this chapter. We refer the reader to [49] and [151] for the detailed results of the implementation.

5.9 Related work

Techniques used to extract data from wireless sensor networks can be classified into three separate categories: (1) *snap-shot* queries, (2) *event-based* queries and (3) *long-running* queries. Snap-shot queries are typically used when the user sends in a query in order to retrieve instantaneous results that reflect the state of the sensors in the network at a certain point in time [60, 126, 72, 70, 54]. Event-based queries on the other hand are completely dependent on the environment that is being monitored, i.e. sensor readings are only transmitted to the sink node if an interesting event has taken place [82, 35, 139, 140, 99]. Readings from long-running queries are obtained using a sampling frequency specified within a query injected into the network by the user [95, 147, 82, 128, 67, 91, 51].

Our application at AIMS specifically requires long-running queries. However, long-running queries can be resolved in various ways. There are long-running queries that extract every single reading acquired by all the sensors in the network. We refer to this as raw data collection. This naturally is not a feasible technique for energy-constrained WSNs due to excessive energy consumption, bottlenecks, reduction in data quality, etc. As these problems were identified in the earlier days of sensor network research, a greater emphasis was placed on in-network processing, i.e. processing the acquired data within the network before transmitting it to the sink node. For example, in Directed Diffusion [82], a node may use a filter to prevent duplicate notifications of an event from being reported numerous times to the sink node. TinyDB [95] and COUGAR [147] on the other hand suggest aggregating data by executing aggregation operators (e.g. MIN, MAX, SUM, COUNT, AVERAGE, etc.) within the network. TiNA [128] presents improvements over TinyDB and COUGAR by taking advantage of temporal correlations of sensor readings.

However, such in-network processing techniques are not suitable for many environmental monitoring projects in general (e.g. our example application at the Great Barrier Reef). The main reason for this is that raw data collection allows *all* the data to be captured and this data can then be analyzed in a variety of ways at a later date. As an example, scientists at AIMS are *not* interested in retrieving the average temperature readings at periodic intervals. Additionally, snapshot queries can always be posed on the raw data that has already been collected. Having all the data enables scientists to interpret the data in whichever way they wish at any time in the future. Other authors in [94, 51] also describe similar scenarios where environmentalists prefer collecting only raw data rather than data that has been manipulated within the network using certain aggregation operators.

One of the ways to perform raw data collection is to take advantage of spatial and temporal correlations of adjacent sensors. This has been done previously in a number of research papers [35, 139, 140, 91]. However, it is important to keep in mind that spatial and temporal correlations that have been identified at time t, may not necessarily hold true at time $t + x$ where $x > 0$. In fact, there may be a situation where two neighboring nodes which usually have correlated readings, may not have correlated readings during certain hours of the day. Thus nodes should be able to adapt their operations accordingly. Though we have not discussed this issue in this chapter with regards to $\mathcal{DOSA}$ (since this chapter focuses only on the scheduling aspects), we would like to indicate that nodes that are unable to find any significant correlations between their adjacent neighbors can actually autonomously opt out of the $\mathcal{DOSA}$ scheduling scheme. Thus a node that chooses to transmit correlation information, only does so if valid correlations exist. The techniques mentioned in [139, 140, 61] are not able to cope with sudden changes in the correlation models and also fail to recognize the importance of temporal fluctuations in these models. Furthermore, the approach presented in [139, 140] is designed for event-based queries. They also assume that individual nodes are location aware. It is important to note that nodes executing $\mathcal{DOSA}$ do not need to be location aware. This definitely reduces the complexity of the software running on the nodes. Unlike $\mathcal{DOSA}$ which is designed for multihop networks, [35, 91, 74] require all nodes in the WSN to be in direct transmission range of the base station. Such a design constraint affects scalability as it prevents these solutions from being implemented in large-scale networks. While Ken [51] takes advantage of spatial and temporal correlations and works in a multihop environment, it does not mention any details of how to re-organize the scheduling scheme if a certain node fails or if new nodes are added to the system. $\mathcal{DOSA}$ on the other hand is able to cope with

network dynamics due to the close interaction that exists with the underlying LMAC layer. The cross-layer optimizations we perform also enable $\mathcal{DOSA}$ to operate in a more energy-efficient manner. PAQ [135] takes advantage of spatial correlations between nodes to reduce transmissions. However, the cluster heads are prone to draining their energy earlier than the cluster members as only the cluster heads are involved in periodic transmission of readings to the sink. While SAF [134] improves on PAQ by ensuring that nodes send "trends" instead of actual sensor readings, it forms clusters off-line and thus fails to take advantage of adjacent nodes that may have correlated sensor readings. Thus all nodes that detect a change in the trend due to some sudden event, are required to transmit model updates to the sink. In $\mathcal{DOSA}$, however, only the correlating node would have to send a model update in such a scenario. Both SAF and PAQ also disregard the underlying MAC completely and are thus unable to benefit from any cross-layer optimizations. The authors of SAF and PAQ also do not provide any theoretical bounds of the energy savings that can be gained using their approach.

While there have been many MAC protocols designed for sensor networks, e.g. S-MAC [148], T-MAC [59], and D-MAC [92] none of the protocols provide neighborhood information the way LMAC does. As shown in our results in Section 5.7.3, the cross-layer optimization we perform using the information presented by LMAC allows us to attain savings of up to 60%. Also, the initial assignment of LMAC slots helps in the second phase of assigning multiple $\mathcal{DOSA}$ colours to a node. The fact that LMAC is a TDMA-based MAC is an added advantage as it automatically provides a sense of time which is beneficial to $\mathcal{DOSA}$. While we have illustrated the operation of $\mathcal{DOSA}$ on top of LMAC it should be possible to run it on top of other MAC protocols. However, this would mean that an addition layer would have to be built which helps keep track of immediate topology information. This would naturally reduce the efficiency of the system.

5.10 Conclusion

The collection of raw sensor readings is of great importance to many applications in sensor networks. We have presented a distributed scheduling algorithm, $\mathcal{DOSA}$ that helps collect raw data in an energy-efficient manner by taking advantage of the spatial correlations that exist between sensor readings of adjacent nodes. Our algorithm is completely self-organizing in the sense that nodes are able to autonomously choose new schedules when there are topology

changes in the network. This is possible due to the close integration with the underlying MAC protocol. This cross-layer approach also results in significant energy savings. We have presented both the theoretical performance bounds and simulation results. Our simulation results indicate a reduction in message transmissions by up to 85% and an increase in network lifetime of up to 92% when compared to collecting raw data. Our algorithm is also capable of completely eliminating dropped messages due to buffer overflow thereby improving the quality of the collected data.

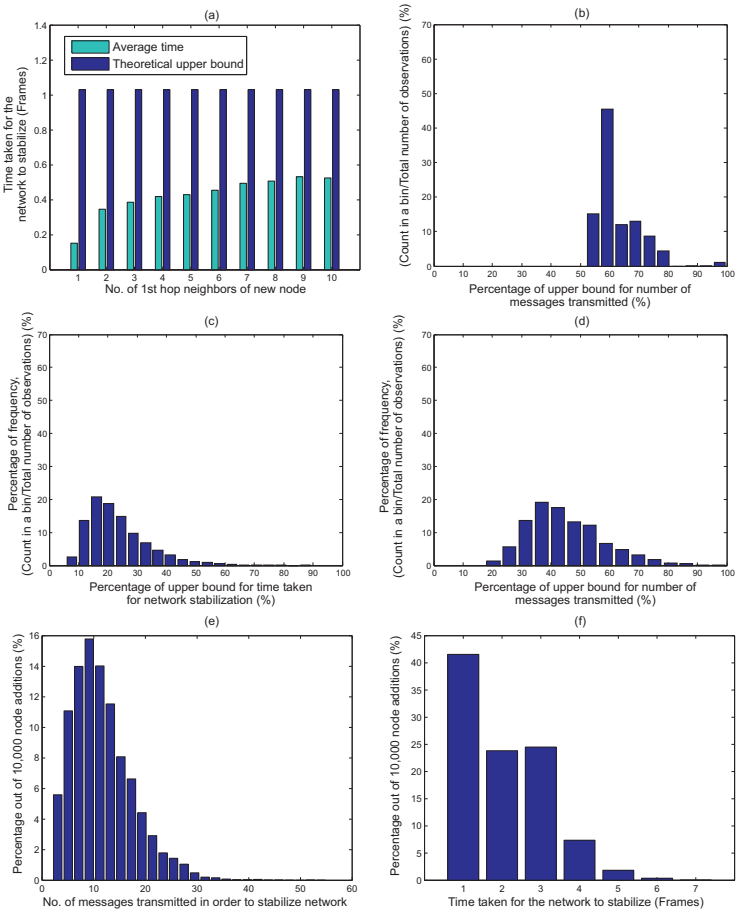

Figure 5.19: (a) Time taken for a network to stabilize once a node has been added to the network, (b) How often the upper bound of the number of messages transmitted for the "1st order colour collision" event is reached when a new node is added, (c) How often the upper bound of the time taken for network stabilization for the "2nd order missing colour" event is reached when a new node is added, (d) How often the upper bound of the number of messages transmitted for the "2nd order missing colour" event is reached when a new node is added, (e) Distribution of number of messages transmitted when a new node is added to the network, (f) Distribution of time taken to stabilize network when a new node is added to the network

Chapter 6

An Adaptive and Autonomous Sensor Sampling Frequency Control Scheme

Collecting raw data from networks used for environmental monitoring can lead to excessive energy consumption. This is especially true when the application requires specialized sensors that have very high energy consumption, e.g. hydrological sensors for monitoring marine environments. Unlike the solution proposed in Chapter 5, which takes advantage of spatial correlations between adjacent sensor nodes, this chapter illustrates how energy can be reduced by taking advantage of temporal correlations between successive readings within a particular sensor node. We describe an adaptive sensor sampling scheme where nodes change their sampling frequencies autonomously based on the variability of the measured parameters[1].

[1]S. Chatterjea, and P. Havinga. An Adaptive and Autonomous Sensor Sampling Frequency Control Scheme for Energy-Efficient Data Acquisition in Wireless Sensor Networks. In *Proceedings of the Fourth IEEE Conference on Distributed Computing in Sensor Systems* (DCOSS), June 2008, Santorini, Greece. Pages to appear. Lecture Notes in Computer Science. Springer Verlag.

The sampling scheme also meets the user's sensing coverage requirements by using information provided by the underlying MAC protocol. This allows the scheme to automatically adapt to topology changes. Our results based on real and synthetic data sets, indicate a reduction in sensor sampling by up to 93%, reduction in message transmissions by up to 99% and overall energy savings of up to 87%. We also show that generally more than 90% of the collected readings fall within the user-defined error threshold.

6.1 Introduction

The primary source of energy consumption in a sensor node is generally attributed to the operation of the radio transceiver [33]. However, certain applications require specialized sensors that have very high power consumption. As an example, in the previously mentioned GBR application (Section 2.3.1), scientists at AIMS require more sophisticated sensors which are capable of monitoring parameters such as salinity or the level of dissolved oxygen [29]. Such sensors can have power consumption levels ranging from 105mW to 720mW. Thus these specialized sensors could typically consume around 30 times more power than a standard sensor node transceiver (e.g Nordic nRF905 consumes $27mW$ in transmit mode [138]). Note also that these sensors commonly have very long start-up and sampling times (e.g. around 2 seconds). Thus in such applications, since sensor sampling operations could have a drastic impact on network lifetime, it is essential to not only reduce usage of the transceiver but also to reduce the number of sensor sampling operations.

In this chapter we take a two-pronged approach to reducing energy consumption by not only reducing sensor sampling operations but also reducing the number of messages transmitted. This is carried out by exploiting the temporal correlations that may exist between successive sensor readings. The basic idea is to use *time-series forecasting* to try and predict future sensor readings. When the trend of a particular sensor reading is fairly constant and thus predictable, we reduce the sensor sampling frequency and message transmission rate. However, when the trend changes, both the sampling rate and message transmission rates are increased.

However, the lower sampling frequency used in such a sampling scheme may result in certain important events being missed. In order to minimize the chance of this from occurring, our sampling scheme tries to ensure an acceptable level of coverage by allowing nodes in the network to adjust various parameters autonomously rather than using fixed values that are predefined prior to deployment. This allows the nodes to operate in a more energy-efficient manner as they are able to automatically adapt their operation to not only the variations in the environment but also to the user-specified coverage requirements. Another novel feature of our scheme is that nodes make use of topology information provided by the underlying MAC protocol (LMAC [78]) in order to adjust the sampling frequency. We are not aware of any existing work where cross-layer optimization is performed between the MAC and application layers. Usually optimizations are restricted to adjacent layers of the OSI model [46]. Our cross-layered approach allows nodes to automatically re-adjust their sampling frequencies if topology

changes are reported by the MAC layer.

We envision our sampling scheme to be used in non-critical applications, e.g. monitoring various parameters in the waters in the GBR, in a coffee beans storage warehouse, etc. This scheme is not meant for critical applications where lives may be at risk if a particular event is missed, e.g. a nuclear plant. In order to evaluate our scheme, we use both real-life and synthetically generated data sets. The real-life data sets are based on temperature readings obtained from outdoor (GBR [37]) and indoor environments (Intel Berkeley Laboratory [22]).

Our contributions are stated as follows:

1. We present a technique to generate synthetic data sets using altitude information and then classify their behavior according to a Variability Index. This helps us evaluate the performance of our sampling scheme under different conditions.

2. We present a localized sensor sampling frequency control scheme that helps reduce sampling operations by up to 93% and transmission operations by up to 99%.

3. We illustrate how a node can adjust certain parameters autonomously in order to ensure coverage is kept at acceptable level across different data sets with different characteristics.

4. We show that the overall energy savings due to our adaptive sampling scheme can be up to 87% while around 90% of the collected data falls within the user-specified threshold.

The following section provides a brief overview of time-series forecasting. This is followed by a description of how we use elevation data to generate synthetic data sets. We then present details of our adaptive sensor sampling scheme and provide the simulation results. Section 6.5 mentions the related work and finally the chapter is concluded in Section 6.6.

6.2 Preliminaries of time-series forecasting

Time-series forecasting is a technique that has been used in a wide variety of disciplines such as engineering, economics, and the natural and social sciences to predict the outcome of a particular parameter based on a set of historical stochastic values. These stochastic values, often referred to as a "time series", are spaced equally over time and can represent anything from monthly sales

data to temperature readings acquired periodically by sensor nodes. William W.S. Wei [141] and Brockwell and Davis [39] provide a very good introduction to time series forecasting.

The general approach to time-series forecasting can be described in four main steps:

1. Analyze the data and identify the existence of a trend or a seasonal component.
2. Remove the trend and seasonal components to get *stationary* (defined below) residuals. This may be carried out by applying a transformation to the data.
3. Choose a suitable model to fit the residuals.
4. Predict the outcome by forecasting the residuals and then inverting the transformations described above to arrive at forecasts of the original series.

Before describing details of how we perform each of the above steps in our data aggregation framework, we first present some basic definitions.

Definition 6.2.1. *Let X_t be a time series where $t = 1, 2, 3, ...$ We define the mean of X_t as,*

$$\mu_t = E(X_t) \tag{6.1}$$

Definition 6.2.2. Covariance *is a measure of to what extent two variables vary together. Thus the covariance function between X_{t_1} and X_{t_2} is defined as,*

$$\gamma(t_1, t_2) = Cov(X_{t_1}, X_{t_2}) = E[(X_{t_1} - \mu_{t_1})(X_{t_2} - \mu_{t_2})] \tag{6.2}$$

Definition 6.2.3. *We define the sample autocovariance at lag h of X_t for $h = 0, 1, 2, ..., T$ as*

$$\hat{\gamma}(h) = \frac{\sum_{t=h+1}^{T}(X_t - \bar{X})(X_{t-h} - \bar{X})}{T} \tag{6.3}$$

where $\bar{X} = T^{-1}\sum_{t=1}^{T} X_t$ is the sample mean of the time series X_t and T refers to the number of observations. Note that $\gamma(0)$ is simply the variance of X_t.

Definition 6.2.4. *The sample* autocorrelation function *(ACF), ρ_h, which indicates the correlation between X_t and X_{t+h}, is*

$$\hat{\rho}(h) = \frac{\hat{\gamma}(h)}{\hat{\gamma}(0)} \tag{6.4}$$

Definition 6.2.5. *We consider the time series X_t to be* stationary *if the following two conditions are met:*

$$E(X_t) = \mu_t = \mu_{t+\tau} \forall \tau \in \mathbb{R} \tag{6.5}$$

$$\gamma(t+h, t) = \gamma(t+h+\tau, t+\tau) \forall \tau \in \mathbb{R} \tag{6.6}$$

Equation 6.5 and Equation 6.6 imply that the mean and covariance for a given lag h remain constant over time respectively.

Definition 6.2.6. *A process is called a* white noise *process if it is a sequence of uncorrelated random variables with zero mean and variance, σ^2. We refer to white noise using the notation WN($0, \sigma^2$). By definition, it immediately follows that a white noise process is stationary with the autocovariance function,*

$$\gamma(t+h, t) = \begin{cases} \sigma^2 & \text{if } h = 0, \\ 0 & \text{if } h \neq 0 \end{cases} \tag{6.7}$$

6.2.1 Analysis of data and identification of trend

As mentioned earlier, the first step is to either identify the trend or seasonal component. However, as we make predictions using a small number of sensor readings taken over a relatively short period of time (e.g. 20 mins), we make the assumption that the readings do not contain any seasonal component. Instead, given that t represents time, we model the sensor readings, R_t using a slowly changing function known as the *trend component*, m_t and an additional stochastic component, X_t that has zero mean. Thus we use the following model: $R_t = m_t + X_t$.

The main idea is to eliminate the trend component, m_t, from R_t so that the behavior of X_t can be studied. There are various ways of estimating the trend for a given data set, e.g. using polynomial fitting, moving averages, differencing, double exponential smoothing, etc. Due to the highly limited computation and memory resources of sensor nodes, we make use of a first degree polynomial, i.e. $m_t = a_0 + a_1 t$.

The coefficients a_0, and a_1 can be computed by minimizing the sum of squares, $Q = \sum_{t=1}^{T} (R_t - m_t)^2$. In order to find the values of a_0 and a_1 that minimize Q, we need to solve the following equations:

$$\frac{\partial Q}{\partial a_0} = -2\sum_{t=1}^{T}(R_t - a_0 - a_1 t) = 0 \tag{6.8}$$

$$\frac{\partial Q}{\partial a_1} = -2\sum_{t=1}^{T}(R_t - a_0 - a_1 t)t = 0 \tag{6.9}$$

Solving equations 6.8 and 6.9 leads to:

$$a_0 = \frac{\sum_{t=1}^{T}(t - \bar{t})(R_t - \bar{R})}{\sum_{t=1}^{T}(t - \bar{t})^2} \tag{6.10}$$

$$a_1 = \bar{R} - a_0\bar{t} \tag{6.11}$$

where $\bar{t}$ and $\bar{R}$ are averages of t and R_t respectively over T observations.

Eliminating the trend component from the sensor readings results in the residuals shown in Figure 6.1(b). The residuals display two distinct characteristics. Firstly, there is no noticeable trend and secondly there are particular long stretches of residuals that have the same sign. This would be an unlikely occurrence if the residuals were observations of iid noise with zero mean. This smoothness naturally indicates a certain level of dependence between readings [39]. Our aim is to study this dependence characteristic which in turn would help understand the behavior of the residuals so that predictions can be made.

Now that a stationary time series has been obtained, the next step is to choose an appropriate model that can adequately represent the behavior of the time series.

Stationary processes can be modelled using *autoregressive moving average* (ARMA) models. The ARMA model is a tool for understanding and subsequently predicting future values of a stationary series. The model consists of an autoregressive part, AR and a moving average part, MA. It is generally referred to as the ARMA(p, q) model where p is the order of autoregressive part and q is the order of the moving average part. The AR(p) model is essentially a linear regression of the current value of the series against p prior values of the series, $X_{t-1}, X_{t-2}, ..., X_{t-p}$. The MA model on the other hand is a linear regression of the current value of the series against the white noise of one or more prior values of the series, $Z_{t-1}, Z_{t-2}, ..., Z_{t-p}$. The complete ARMA$(p, q)$ model is defined as follows,

$$X_t = \phi_1 X_{t-1} + ... + \phi_p X_{t-p} + Z_t + \theta_1 Z_{t-1} + ... + \theta_q Z_{t-q} \tag{6.12}$$

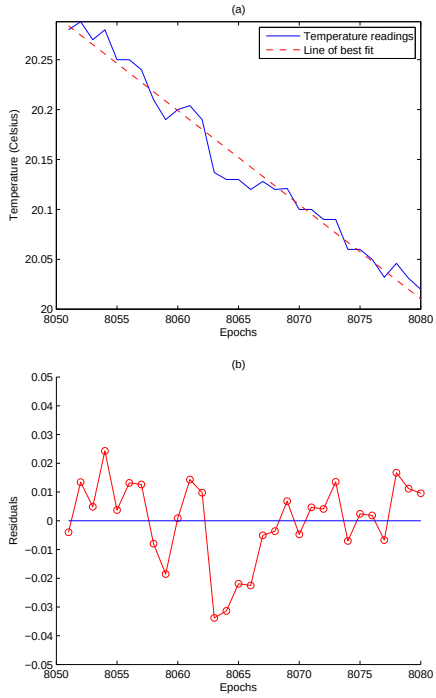

Figure 6.1: Temperature sensor readings and the corresponding residuals (Data obtained from Nelly Bay in the GBR [37])

where $Z_t \sim \text{WN}(0, \sigma^2)$ and ϕ_i, $i = 1, 2, ..., p$ and θ_i, $i = 1, 2, ..., q$ are constants [39].

However, due to the limited computation and memory resources on a sensor node, we use an AR(1) model instead of the full ARMA model (i.e. $q = 0$) to predict the value R_t, i.e. $X_t = \phi_1 X_{t-1} + Z_t$. The constant ϕ_1 can now be estimated using the Yule-Walker estimator, i.e. $\hat{\phi_1} = \hat{\rho_1} = \frac{\hat{\gamma}(1)}{\hat{\gamma}(0)}$ [141]. We can then state, as shown in [141], that the general form of the minimum mean square error m-step forecast equation is

$$\hat{X}_{t+m} = \mu + \hat{\phi}^m (X_t - \mu), m \geq 1 \tag{6.13}$$

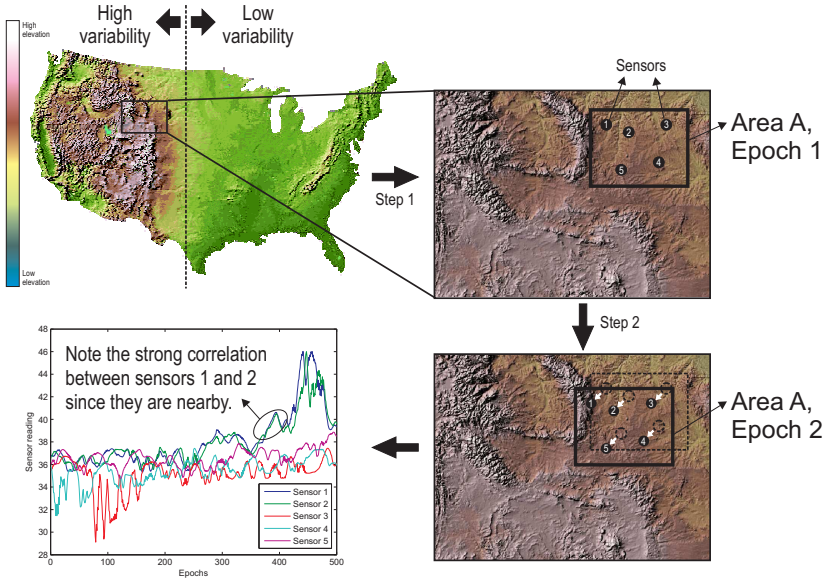

Figure 6.2: Overview of how a synthetic data set is generated

6.3 Data set generation and classification

The algorithms presented in this chapter are designed specifically for sensor networks that are deployed for environmental monitoring purposes. In order to extensively evaluate the performance of the various algorithms, it is essential to test them using several data sets that display different characteristics, e.g. while the sensor readings in one data set may be relatively constant, the readings in another data set may change rapidly. Unfortunately, due to the lack of large scale sensor network deployments, large, real-life data sets are still not readily available. While there are a few small real-life data sets available [22], they do not display the wide range of characteristics that is required for having a complete evaluation of the performance of the algorithms.

In order to circumvent this problem of the unavailability of certain types of data sets, we now present a technique to not only generate different data sets but also describe how the different data sets can be classified based on their variability. The technique allows the generation of data sets that are correlated both spatially and temporally.

We generate synthetic sensor readings using elevation data obtained from the Seamless Data Distribution System provided by the U.S. Geological Survey (USGS) [20]. The site provides elevation data of the entire United States up to a resolution of 10 meters. The first step is to choose an appropriate area on the map. The area chosen depends on the required variability of the synthetic data set. For example, if the synthetic data set needs to have high variability, one may opt to choose the western part of the United States which is generally more mountainous, i.e. the *variation* of elevation changes significantly within a small distance. Conversely, if the user requires sensor readings where the changes are fairly small, one may choose an area that lies in the central part of the map. This is illustrated in Step 1 in Figure 6.2. Next an area is chosen within the area selected in the previous step. This area is labeled as A in Figure 6.2. Area A indicates the size of the deployment area of the sensor network as specified in the simulations. Nodes are then positioned randomly within this area and elevation readings are obtained at the location of every node. Thus if there are 100 randomly deployed nodes, a total of 100 elevation readings will be obtained and these are taken to be the sensor readings for the first epoch. Note that readings between nodes situated close to one another will be spatially correlated. Next, area A is shifted one unit in a particular direction together with all the nodes within it. New elevation readings are once again obtained at the location of every node. These readings are assumed to be the sensor readings for the second epoch. Thus readings of a particular node will be temporally correlated over the various epochs. This process is repeated until readings for the required number of epochs is obtained. The graphs in Figure 6.2 also show how the generated synthetic data set has both spatial and temporal correlations.

Another inherent property of data sets generated using this technique is the fact that different areas of the deployment area may have different levels of variation over a fixed duration. This not only mimics the characteristics of real-life data sets but is also very useful for testing the localized behavior of nodes, e.g. while some nodes in the network may experience high variability in the sensed readings, other nodes may experience low variability. Thus in order to operate in an optimized manner it is essential for the algorithms running on the nodes to adapt to the nature of the data that is being sensed. Figure 6.3 illustrates how both a real life data set (obtained from Intel Berkeley [22])and synthetic data sets display different levels of variability in sensor readings in different regions. Note that because the performance of the algorithms described in this chapter depend on the *rate of change of the trend* of a sensor reading and not the sensor reading itself, the contour maps in Figure 6.3 indicate *how often the trend* of the sensor readings changes.

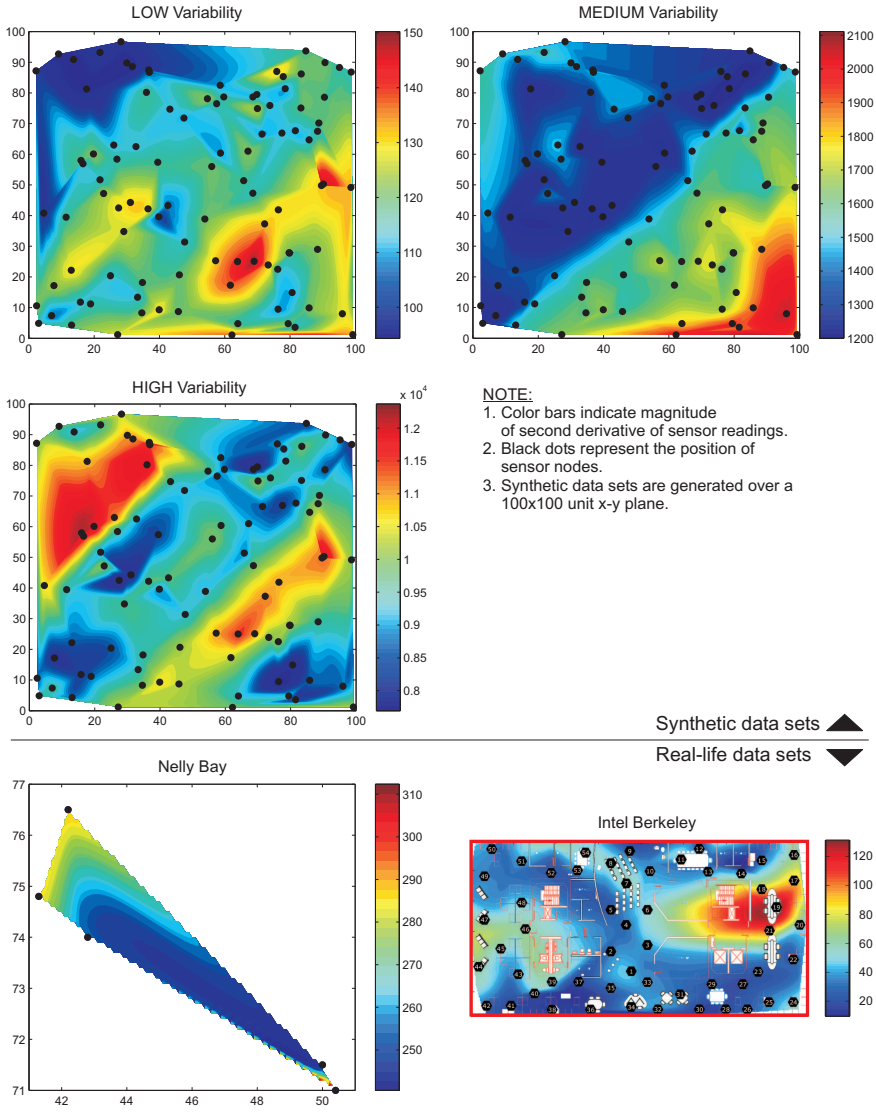

Figure 6.3: Variability of the various real and synthetic data sets

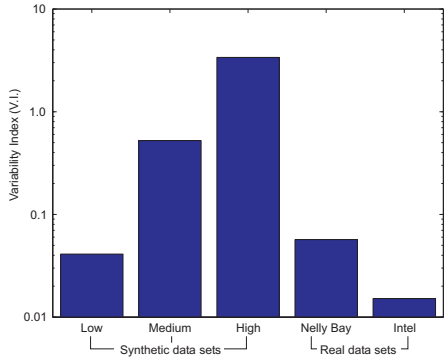

Figure 6.4: Variability Indices of different synthetic and real-life data sets

In order to test our algorithms extensively under different conditions, we shall use three different data sets which have varying levels of variability. The graph in Figure 6.4 shows how we classify data based on a *Variability Index* (V.I.). The V.I. shows the *average rate of change of the trend of a sensor reading per epoch per node*. For the sake of comparison, we have also included data obtained from two real life data sets: (i) temperature data from Nelly Bay in the GBR [37] and (ii) temperature data from the Intel Berkeley Lab [22]. It can be seen that the real-life data set clearly falls within the *low variability* category. We have chosen this metric to classify different data sets as the performance of our approach depends on how often the trend of a sensor reading changes.

6.4 Localized sensor sampling frequency control

As mentioned earlier, apart from the operation of the radio transceiver, the sampling of the various sensors on a single node may also consume a large amount of energy. In this section we describe a local adaptive sensor sampling frequency mechanism where the sensor sampling frequency depends not just on the predictability of the physical parameter being measured but also on the user-specified error threshold.

In general terms, when the reading of a particular sensor on a node can be predicted based on the recent past, we reduce the frequency of sampling the sensors by skipping a number of sensor sampling operations and performing predictions instead. However, the moment the prediction differs from the actual

sampled reading by an amount specified by the user, the sampling frequency is increased. This local prediction mechanism also helps reduce the number of sensor readings that need to be transmitted to the sink node. We now describe the precise steps in greater detail.

After all nodes in the network have acquired LMAC slots, a query is injected into the network specifying the required time interval (e.g. t time units) between successive sensor samples. The query also describes the error levels($\pm\delta$ units) that may be tolerated by the user. Upon receiving a query, all nodes initially acquire the first r readings at intervals of t time units and store these readings together with their corresponding sample times in the buffer. Note that the maximum number of (time, sensor reading) tuples that can be stored in the buffer is r. Once the first r readings have been acquired, the reading for epoch $(r+1)$ is not only acquired but also predicted. We refer to the nth reading *acquired* after the r readings in the buffer as $R_{A(r+n)}$ and the nth reading *predicted* after the r readings in the buffer as $R_{P(r+n)}$. The method used to carry out the predictions is described in Section 6.4.1.

If $|R_{P(r+1)} - R_{A(r+1)}| \leq \delta$, it means that the prediction falls within the user-specified error margin. Since the prediction is accurate enough, we make the assumption that the prediction accuracy will continue to hold for the following epoch $(r+2)$. Thus the node skips acquiring a sample in epoch $(r+2)$. This is done by setting the variable `CurrentSkipSamplesLimit` (`CSSL`) to 1. `CSSL` indicates the number of samples that need to be skipped before the next sample is taken. The variable `SkipSamples` (`SS`) is then set to the value held by `CSSL`. For every sample skipped, the value of `SS` is decremented by 1. Every time `SS` reaches a value of 0, the node samples the sensor to acquire a reading and a prediction is also computed. Thus in this case, `SS` reduces to 0 in epoch $(r+3)$ and a sensor reading, $R_{A(r+3)}$ is acquired and $R_{P(r+3)}$ is calculated. This time, if $|R_{P(r+3)} - R_{A(r+3)}| \leq \delta$, `CSSL` is incremented by one (i.e. it is set to 2) and `SS` is then set to the value held by `CSSL`. In other words, every time an accurate prediction is made, `CSSL` is incremented by 1 and `SS` decreases to 0 from an initial starting value of `CSSL`. If however, at any time $|R_{P(r+3)} - R_{A(r+3)}| > \delta$, i.e. the prediction made is inaccurate, both `CSSL` and `SS` are set to zero so that no samples are skipped. Since the node samples its sensor and compares it with the prediction based on the past r readings stored in its buffer every time `CSSL` and `SS` are set to 0, a node can once again start skipping samples the moment it detects that the sensor readings can be predicted accurately.

However, if a node continues to make a long line of correct predictions, there is the possibility of `CSSL` increasing infinitely. This of course is undesirable as it would mean that eventually, when a change in the measured parameter does

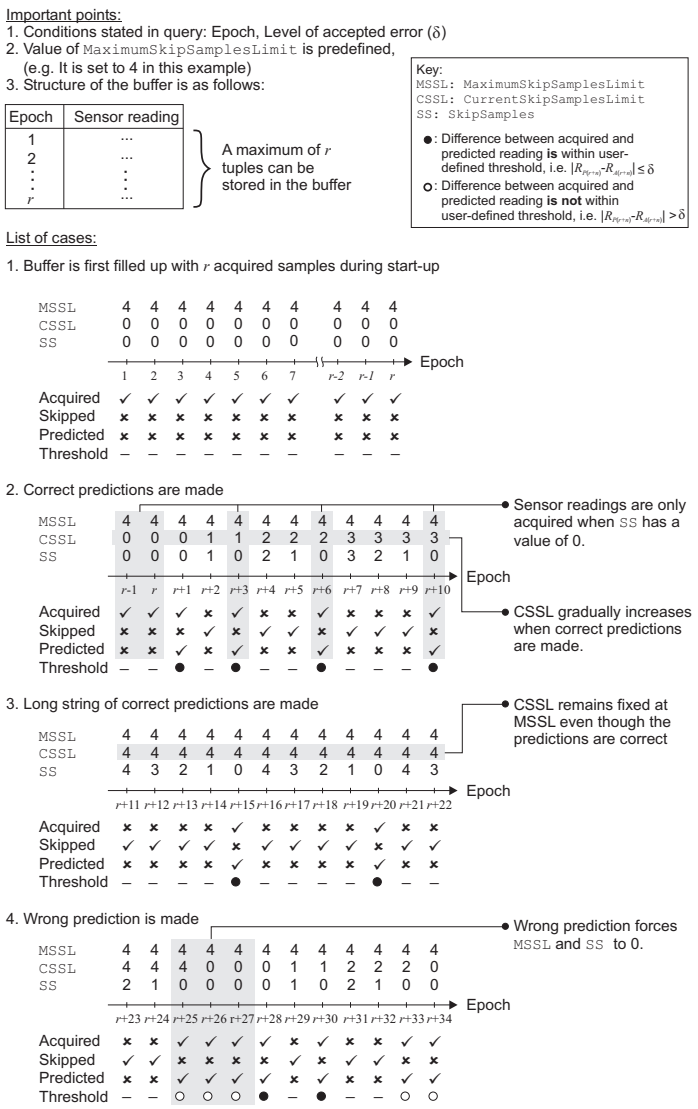

Figure 6.5: Example illustrating how the values of the various variables change

take place, it would be impossible to detect it. Thus we set a maximum limit to the value of `CSSL` known as the `MaximumSkipSamplesLimit` (`MSSL`). Once this maximum limit is reached, the variable `CSSL` continues to remain at that limit even if further correct predictions are made. Figure 6.5 summarizes the method presented above with an example illustrating how the values of `CSSL` and `SS` vary. While we initially use static values of the various variables to illustrate the benefits of this technique, in Section 6.4.1.3 we describe how a node may autonomously adjust the values of the variables as the conditions in the environment and network change.

6.4.1 Prediction of sensor readings

As mentioned earlier, we make use of time series forecasting to predict sensor readings. In basic time series forecasting, the historical values based on which predictions are supposed to be made, *must* be equally spaced. However, in the approach presented above it is evident that sensor readings are not acquired at regular intervals, e.g. Figure 6.5 clearly shows that an increasing number of samples are skipped when several correct predictions are made consecutively. While there are ways to estimate missing observations, such strategies would not only involve additional computation but may also not be very effective. This is because in our sampling scheme, in many instances, the majority of sensor readings may be missing (e.g. a buffer which is capable of storing 20 readings, may contain 18 missing readings and only 2 sampled readings).

As shown in Figure 6.6, every prediction is made based on the most recent r readings (representing the previous r epochs) stored in the buffer. Note that *all* the r readings are used regardless of whether they are sampled or predicted readings. Since the initial prediction is made for the epoch that *immediately* follows the epoch in the r^{th} position of the buffer, the prediction is made using Equation 6.14 and setting m to 1 (i.e. we carry out a 1-step forecast), i.e.

$$\hat{X}_{t+1} = \mu + \phi(X_t - \mu) \tag{6.14}$$

Once the prediction is made, the first reading in the buffer is removed. The remaining readings are then moved back one position while the newly predicted reading (and the corresponding epoch) is placed in the r^{th} position of the buffer. This process is repeated for all subsequent predictions (e.g. when `SS`>1) until a sample is taken. When a sample is taken, as shown in Figure 6.5, it is always compared with a prediction. However, instead of now placing the predicted reading in the r^{th} position of the buffer, the actual sampled reading is placed

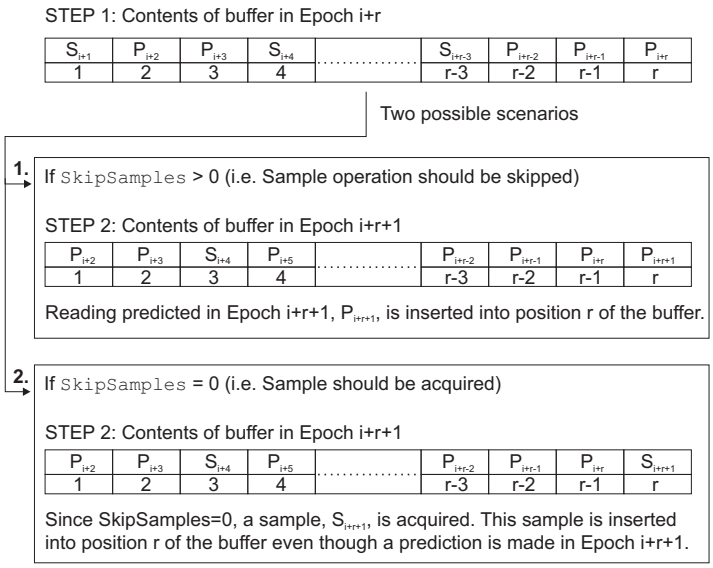

Figure 6.6: Updating the buffer

there instead. It is obvious that the greater the number of sampled readings, the more accurate the predictions will be. Thus inserting every sampled reading into the buffer ensures that the maximum possible advantage is derived from the energy spent sampling the sensor. This is illustrated in Figure 6.6.

6.4.1.1 Transmission of model updates to sink node

The final aim of any data collection system is to transmit the requested data to the user - who we assume in this chapter to be located at the sink node. In a conventional data collection system, every node in the network would transmit a reading at every epoch (specified by the user) towards the sink node. This naturally not only results in excessive power consumption but also results in poor data quality since nodes closer to the root node may drop messages due to buffer overflows [48].

Our presented approach not only focuses on reducing the number of samples acquired but also uses the same prediction mechanism described in Section 6.4.1 to reduce the number of messages that need to be transmitted to the sink node. In short, instead of transmitting every acquired reading, or transmitting every

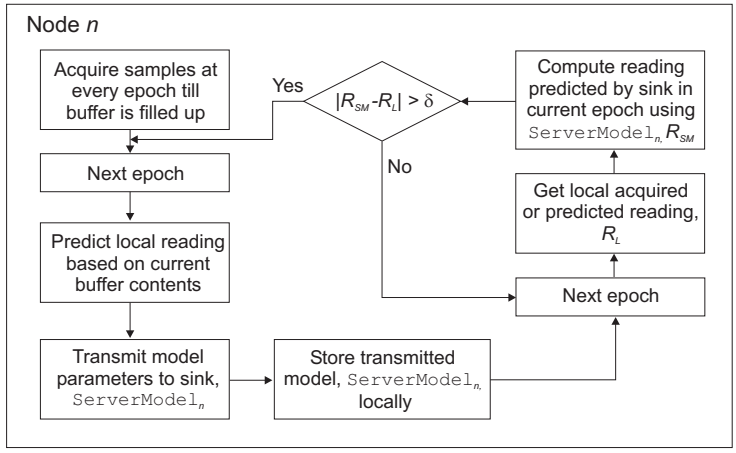

Figure 6.7: Flowchart illustrating when model updates are transmitted to the sink by node n

time there is a change greater than the user-specified δ, we only transmit when there is a significant change in the *trend* of the measured parameter.

When the system first starts up, as mentioned earlier, a node n first samples its sensor in every epoch until the buffer gets filled up. A node does not transmit these readings to the sink node. As mentioned earlier, the first prediction is made immediately after the buffer has filled up and is based on all the inputs in the buffer. The parameters used to carry out the prediction (i.e. a_0, a_1, t, ϕ_1, X_t) are then transmitted to the sink. We refer to these model parameters of node n as $\texttt{ServerModel}_n$. Node n also stores $\texttt{ServerModel}_n$ locally. The sink node uses the received $\texttt{ServerModel}_n$ parameters to predict future sensor readings of node n. At every epoch after the transmission of the first $\texttt{ServerModel}_n$, node n uses its locally saved copy of $\texttt{ServerModel}_n$ to compute the sink node's prediction of node n's reading at the current epoch. If n computes that the reading predicted by the sink node differs from n's own reading by an amount greater than the user-specified δ, node n then transmits an updated $\texttt{ServerModel}_n$ to the sink. This update is based on the current contents of node n's buffer. In summary, node n transmits an updated $\texttt{ServerModel}_n$ every time the sink node's copy of the previously received $\texttt{ServerModel}_n$ is unable to predict node n's sensor readings accurately. This whole process of transmitting server model updates is summarized in Figure 6.7(b).

Data set	No. of nodes	No. of epochs	V.I.	Total no. of samples acquired		
				Without prediction	**With** prediction, MIN (**% reduction**)	**With** prediction, MAX (**% reduction**)
LOW	100	2880	0.0411	288000	16401 (**93.4**)	98101 (**65.9**)
MEDIUM	100	2880	0.5237	288000	101387 (**64.8**)	165636 (**42.5**)
HIGH	100	2880	3.3769	288000	217451 (**24.5**)	253592 (**13.6**)
NELLY	8	4936	0.0569	39488	2934 (**92.6**)	7881 (**80.0**)
INTEL	51	2880	0.0152	146880	10731 (**92.7**)	50323 (**65.7**)

Table 6.1: Simulation settings and summary of results (Total number of samples acquired) for the static CurrentSkipSamplesLimit scheme (The figures in brackets indicate the percentage reduction when compared to raw data collection scheme)

6.4.1.2 Effect of buffer size and MaximumSkipSamplesLimit

Values chosen for the buffer size and the `MaximumSkipSamplesLimit` (`MSSL`) can have a significant effect on the total number of samples acquired and model transmissions made by a node. In this subsection, we describe the effects of these two parameters for data sets having different V.Is. Simulations were performed using different data sets to investigate how the nodes responded when different values were used for the buffer size and `MSSL`. The simulation settings and results are summarized in Tables 6.1 and 6.2 and the detailed results are presented in Figures 6.8 and 6.9. It can be easily observed that the results from the real data sets from Nelly Bay (Figure 6.9(a),(c)) and Intel (Figure 6.9(b),(d)) look very similar to the Low V.I. results (Figure 6.8(a),(d)) since they belong to the same class as far as the V.I. is concerned. Over the next few paragraphs we provide a detailed explanation of the various characteristics of the graphs in Figures 6.8 and 6.9.

Generally, when the trend of the sensor reading is fairly constant or predictable (i.e. Low V.I.), the `CSSL` will always reach the `MSSL` and remain there regardless of the value of `MSSL`. This is because a constant trend generally results in a long sequence of consecutive predictions which meet the users accuracy requirements. Note that `MSSL` can have a maximum value that is 2 less than the size of the buffer. A large `CSSL` effectively means that a node will sample its own sensor less frequently. Thus for data sets with a low V.I., only the value of `MSSL` affects the number of samples a particular sensor acquires. The size of

Data set	No. of nodes	No. of epochs	V.I.	Total no. of transmissions made		
				Without prediction	**With** prediction, MIN (**% reduction**)	**With** prediction, MAX (**% reduction**)
LOW	100	2880	0.0411	288000	2326 (**99.2**)	4088 (**98.6**)
MEDIUM	100	2880	0.5237	288000	87170 (**69.7**)	150162 (**47.9**)
HIGH	100	2880	3.3769	288000	207896 (**27.8**)	241830 (**16.0**)
NELLY	8	4936	0.0569	39488	2629 (**93.3**)	7298 (**81.5**)
INTEL	51	2880	0.0152	146880	4667 (**96.8**)	12868 (**91.2**)

Table 6.2: Simulation settings and summary of results (Total number of transmissions made) for the static CurrentSkipSamplesLimit scheme (The figures in brackets indicate the percentage reduction when compared to raw data collection scheme)

the buffer does not have any impact.

However, the situation varies for the transmission of model updates. Recall that in the case of predicting readings used to decide on the value of `CSSL`, a node uses the current contents of its own buffer, which is always up-to-date since it relies on a sliding window containing the most recent r readings where r is the size of the buffer. On the other hand, when deciding whether to transmit a model update, the predictions are made using the *previously* transmitted `ServerModel`. Thus predictions for model updates are based on "old" data. In this case, in addition to `MSSL`. the buffer size also has an impact on the number of transmitted model updates as a larger buffer size means that there are a larger number of "older" readings in every transmitted `ServerModel`.

For data sets with a higher V.I. the buffer size plays a more significant role than the choice of the `MSSL` - except when we consider smaller values of `MSSL` (Figures 6.8(b),(c),(e),(f)). Generally, `MSSL` hardly has a role primarily because when the trend is continuously changing, `CSSL` can only take up small values even if `MSSL` is set to a large value. Reducing the size of the buffer size has two opposing effects. Firstly, a small buffer size implies that `MSSL` must also have a small value since `MSSL` can have a maximum value that is 2 less than the buffer size. This in turn means that a node will have to sample its sensor more often. Conversely, when the trend is highly variable, a smaller buffer size would obviously help model the change in trend more accurately thus resulting in more accurate predictions. A smaller buffer size also means that only the recent past is being used for prediction. This means that the a node will sample its sensor

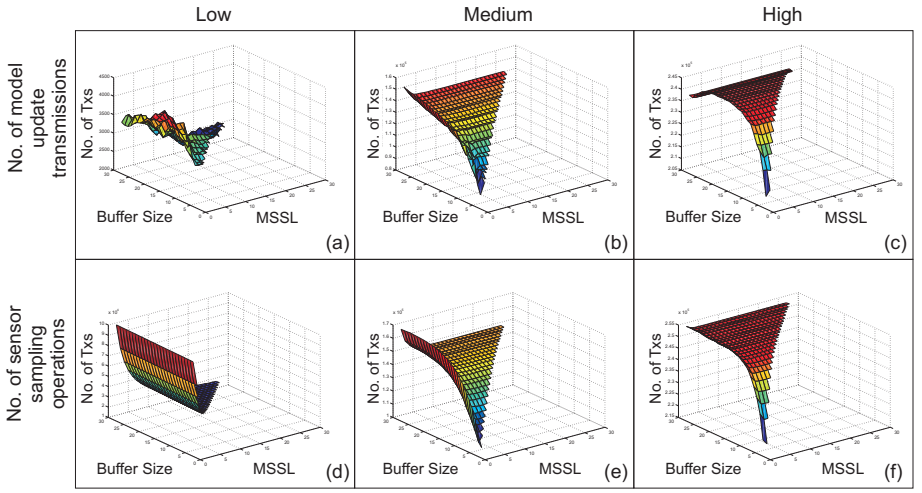

Figure 6.8: Sensor sampling and model transmission results for different *synthetic* data sets using different buffer sizes and MaximumSkipSamplesLimit (MSSL) values (Note: Without the prediction mechanism, the number of sensor sampling and transmission operations for the synthetic data sets would be as follows - Low, Medium, High: 288000)

less often. Our simulation results clearly indicate that when a smaller buffer size is used, the benefits gained from better prediction far exceeds the disadvantage due to more frequent sensor sampling. Since a larger buffer size generally results in more transmissions (which in turn means more accurate data collection at the expense of higher energy consumption) we have used a buffer size of 30 for all the following simulations.

6.4.1.3 Meeting the user's coverage requirements (Dynamic MSSL)

In this sub-section we see how a node can autonomously adjust its `MSSL` in order to meet the coverage requirements specified by the user. We first provide our definition of *coverage* and then describe the mechanism in greater detail.

When `MSSL` is set to m, it means that x unsampled epochs lie between any two consecutive samples. The duration of an event could span over several

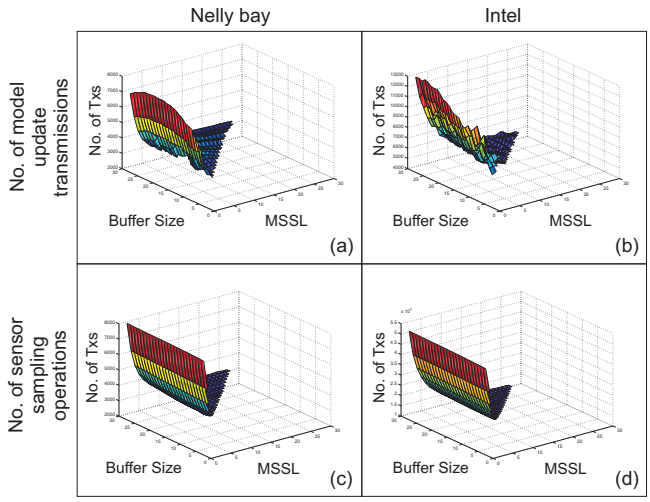

Figure 6.9: Sensor sampling and model transmission results for different *real* data sets using different buffer sizes and MaximumSkipSamplesLimit (MSSL) values (Note: Without the prediction mechanism, the number of sensor sampling and transmission operations for the different real data sets would be as follows - Nelly: 39488, Intel: 146880)

epochs. We consider an event to be detected as long as the sensor samples during *any one* of the epochs during which the event has occurred, i.e. our aim is not to detect an event at the earliest possible time, but to simply detect *if* at all an event has occurred. Note that this is adequate since our solution is meant only for non-critical monitoring applications.

So given that `MSSL` is set to m, the duration of the shortest possible event of interest is d epochs we compute the probability of an event being missed by a particular node, p as $p = \frac{m+1-d}{m+1}$. Note that the denominator, $m+1$, is the total number of positions where an event could occur and the numerator, $m+1-d$, refers to the number of uncovered event positions. Thus *coverage* is defined as the probability that an event *will* be detected, i.e. *1-p*.

If the current node has n neighbors, then we assume that an event is detected as long as any *one* of the nodes within a node's closed 1-hop neighborhood samples its sensor at any time during the occurrence of the event. The reason

for this approach is that the spatial correlations between neighboring sensors in a high density network might mean that if a particular event is detected by one node, there is a high possibility of the same event being detected by an adjacent neighbor (though of course the magnitude of the measured parameter might be different). We refer the reader to our earlier work in [48] which describes typical correlation levels between adjacent sensors in a real deployment in the GBR. We also use this approach for a "distributed wake-up" mechanism where a node that detects an event asks all its adjacent neighbors to break out of the reduced sampling scheme and immediately sample a reading (by reducing SS to 0) to check if they too can detect the event. This would help improve the accuracy of the data collected. However, we do not describe this mechanism here as it falls outside the scope of this chapter.

Thus when multiple nodes are involved in checking the occurrence of a certain event, the probability of missing the event is reduced. This is reflected in the following equation that computes the probability an event is missed when a node has n adjacent neighbors: $p = (\frac{m+1-d}{m+1})^{n+1}$.

Assuming that a user specifies the coverage he or she is willing to tolerate through a query injected into the network, every node can then compute the corresponding `MSSL` based on the number of neighbors a node has by simply rearranging the previous equation:

$$\texttt{MSSL} = \lfloor \frac{\sqrt[n+1]{p} + d - 1}{1 - \sqrt[n+1]{p}} \rfloor \tag{6.15}$$

Figure 6.10 shows a plot of the above equation given that $d = 5$ and the size of the buffer is 30. It can be seen that for any user-specified p, the larger the number of neighbors a node has, the larger is the chosen value of `MSSL` thus allowing a node to spend more energy sampling only when the user requires it. If the computed value of m is greater than 2 less than the buffer size, m is set to $BufferSize - 2$. This explains the flat portion of the graph.

In Figures 6.11(a)-(c) we illustrate how the probability of missing an event varies for different event durations when data sets with different V.Is are used. Note that in the simulations, we assume that the user has predefined p to be 0.2. In Figure 6.11(a), when a static value of `MSSL` is used (e.g. 15, 28), the actual p can exceed the user-specified p. However, this is prevented in the case of Dynamic `MSSL`. Data sets with Medium or High Variabilities do not really require the Dynamic `MSSL` mechanism as `CSSL` values never reach `MSSL` due to the frequent change in trends. Since a small `MSSL` implies that the nodes are sampling very frequently, they hardly ever miss any passing event. This can

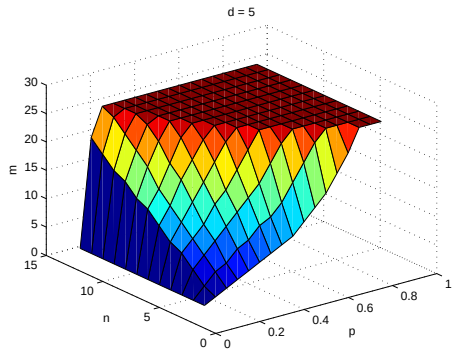

Figure 6.10: Computation of MSSL given p (user-specified) and n (topology dependent). Note that MSSL is set to m.

be clearly observed from Figures 6.11(b)-(c). Thus using the Dynamic `MSSL` mechanism is advisable if a user expects to gather data which has a low V.I. and requires a coverage guarantee.

We evaluate the overall performance of our adaptive sampling scheme by investigating the overall energy consumption of all the nodes in the network and the accuracy of the collected data. We compare our scheme with conventional raw data collection where a reading is collected from every node in every epoch and also with *approximate caching* [113]. In this algorithm, a node transmits a reading to the sink whenever its current value differs from the previously transmitted reading by an amount greater than the user-specified threshold. Note however, that while approximate caching can help reduce the number of transmissions, it cannot be used to reduce the number of sampling operations since the algorithm does not perform any predictions. Our simulations use LMAC. We use 32 slots per frame where each frame is 8 seconds long. The energy model based on the RFM TR1001 transceiver [27] and the EXCELL Salinity Sensor [29] which could typically be used in our GBR deployment. Measurements are required every 30 seconds. The number of nodes used is specified in Table 6.1. All results have been collected over 400 frames or 53.3 minutes.

Figure 6.12(a) shows the energy spent operating the transceiver by all nodes in the network over the 400 frame duration. Even though there is a significant reduction in the number of messages transmitted (as shown in Table 6.2), the

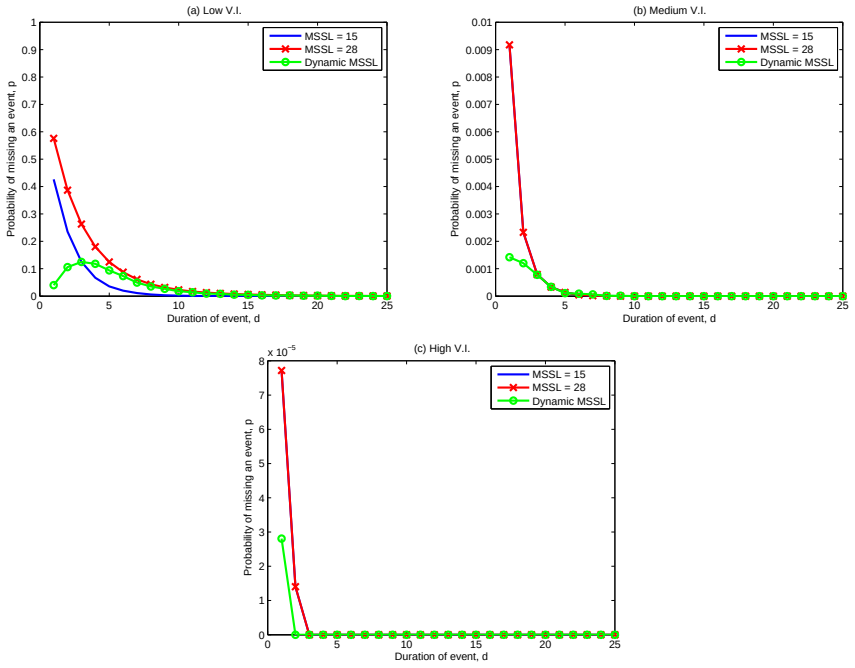

Figure 6.11: Probability of missing an event for different event durations and data sets with different V.Is

energy different between the raw, approximate caching and our adaptive sampling techniques is not very large. This is largely due to the LMAC protocol where nodes spend a lot of time listening in order to maintain synchronization even when there is no data to send. However, as shown in Figure 6.12(b), if we look at the overall energy consumption - which includes both the operation of the transceiver and the sensors - our scheme reduces overall power consumption by up to around 87% (Low V.I.) as compared to the other two schemes. Even though there is such a large reduction in energy consumption, in most cases, more than 90% of the collected readings are within the ± 0.1 units of the actual readings. This is shown in Figure 6.12(c).

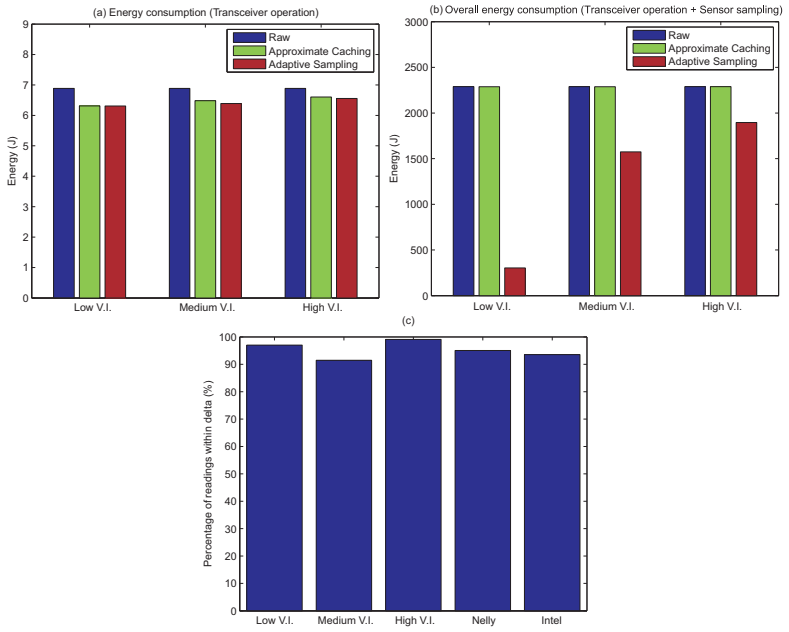

Figure 6.12: (a) Energy consumption due to transceiver operation, (b) Overall energy consumption due to transceiver and sensor operation, (c) Accuracy of collected data

6.5 Related work

A wide variety of techniques can be found that deal with extracting data in an energy-efficient manner. The authors in [60] describe a technique to prevent the need to sample sensors in response to an incoming query. However, the technique is not able to cope with sudden changes in the correlation models and also fails to recognize the importance of temporal fluctuations in these models. Ken [51] is able to adapt create models on the fly and thus adapt but does not describe how to deal with topology changes. Both PAQ [135] and SAF [134] are similar to our scheme in the sense that they also use time-series forecasting to identify temporal correlations within the network itself. However, all the schemes mentioned above only deal with reducing message transmissions. While this is helpful, our results in Figure 6.12(b) clearly indicate that a large reduction in message transmissions does not always translate into energy savings of the same magnitude largely

due to the overhead of the radio (which is dependent on the MAC). This is especially true when the application uses sensors which consume a lot of energy. Note that many sophisticated sensors used for environmental monitoring also have long start-up and sampling times. This too has a large impact on energy consumption. To our knowledge, we are not aware of any other work which deals with reducing the duty cycle of the sensors themselves.

6.6 Conclusion

We have presented an adaptive sensor sampling scheme that takes advantage of temporal correlations of sensor readings in order to reduce energy consumption. Our localized scheme, which uses cross-layer information provided by the MAC, allows nodes to change their sampling frequency autonomously based on changes in the physical parameters being monitored or the topology of the network. Our simulations based on real and synthetic data sets, indicate a reduction in sensor sampling by up to 93%, reduction in message transmissions by up to 99% and overall energy savings of up to 87%. We also show that generally more than 90% of the readings are within the user-defined error threshold.

The scheme presented here does not consider any uncertainties that may be present in the sensor readings. For example, sensor readings may be prone to systematic errors. The effect of uncertainty was not studied here as it was not found to be a significant issue in either of the real-life data sets that we used. However, if the magnitude of systematic errors is considerable, this would decrease the effectiveness of the adaptive sampling scheme, i.e. a greater number of sampling operations and transmissions would be required. In order to reduce the magnitude of uncertainty, it would be necessary to have additional algorithms that pre-process the sampled data to minimize systematic errors before passing the the readings to our adaptive sampling scheme, e.g. calibration algorithms, etc. We will deal with the issue of tackling uncertainty of sensor readings in the future.

Chapter 7

Conclusion and Future Work

This chapter summarizes the primary contributions of this thesis. We present the extent to which the research question mentioned earlier in Chapter 1 has been answered. The thesis is finally concluded by outlining future research directions.

7.1 Overview of contributions

In Chapter 1 we stated that the main research question of this thesis was to investigate how data could be extracted efficiently from a wireless sensor network.

The term "efficient" can be interpreted in a multitude of ways depending on the emphasis placed by the application being considered. As the majority of WSN-related research focuses primarily on techniques to extend the lifetime of the network, a significant proportion of this thesis is dedicated to devising strategies that would help achieve this very goal. However, it is important to keep in mind that network lifetime is not the only issue that is of concern to the end-user. In order for WSNs to be a sound business proposition, it is essential for a deployed sensor network to eventually help solve the problem it was actually designed for. In other words, only having a network which is capable of running continuously for decades without satisfying the user's data quality requirements is definitely not acceptable. Thus the research approach of this thesis embodies the term efficiency with two facets: energy-efficiency and data quality. (Note that data quality could refer to parameters such as packet loss or latency depending on the application requirements.) All the approaches presented in this thesis address both these facets.

This thesis has taken a multi-pronged approach to address the issue of efficient data extraction from sensor networks. Instead of focussing only on a single component of the WSN architecture, we have presented a new cross-layered approach which maximizes information re-use thus improving efficiency. We illustrate how information generated by a particular layer of the sensor network protocol stack can be used by the other layers to help improve their operation as well. To illustrate how the proposed cross-layered approach can be used, we present solutions where the MAC, routing, data aggregation and sensor sampling operations benefit by using cross-layer information.

The data extraction strategies presented in this thesis have been designed for two types of applications:

- Applications where range queries are injected into the network: this is suitable when the user does not require continuous streaming data for long durations from the entire network but only requires data from certain parts where certain conditions have been met. These queries either provide the user with a snap-shot of the various physical parameters being monitored or provide streaming data for a short duration.
- Applications where long running queries are injected into the network: this is used when a continuous stream of data is required from the entire

network over long durations.

We now list the main contributions and results of this thesis:

1. *A cross-layered approach (Chapter 1):* We define a new approach for WSNs where information is exchanged between all the layers above the MAC in order to improve operating efficiency.

2. *A taxonomy of distributed query management techniques (Chapter 3):* We describe and compare various existing query management techniques for WSNs. The drawback of the existing techniques motivates the development of cross-layered approaches for the routing and MAC layers.

3. *A directed query dissemination scheme, DirQ (Chapter 4):* Instead of flooding a network with queries, DirQ is a routing mechanism that routes queries to the appropriate source nodes based on both static and dynamic-valued attributes such as sensor types and sensor values. The routing algorithm is able to automatically adapt to node failures or additions by using cross-layer information acquired from the underlying MAC protocol. The results indicate that the cost of the DirQ scheme lies between 45% and 55% that of tree-based flooding.

4. *An adaptive, information centric and lightweight MAC protocol, AI-LMAC (Chapter 4):* This is a modification of the existing LMAC protocol. However, instead of assigning the same amount of bandwidth to all the nodes in the network, AI-LMAC uses a Data Distribution Table to capture information about the sensed data and uses this data in combination with the requirements of the injected query to decide how much bandwidth each node should allocate. The results indicate how AI-LMAC efficiently manages the issues of fairness and latency.

5. *A distributed and self-organizing scheduling algorithm for energy-efficient data aggregation (Chapter 5):* We present a distributed and self-stabilizing algorithm that aggregates data by taking advantage of spatial correlations that may exist between the sensor readings of adjacent sensor nodes. This algorithm is primarily designed for collecting data in response to long running queries. Our results indicate a reduction in message transmissions of up to 85% and an increase in network lifetime of up to 92% when compared to collecting raw data. The algorithm is also capable of completely eliminating dropped messages due to buffer overflows.

6. *An adaptive and autonomous sensor sampling frequency control scheme (Chapter 6):* Instead of focusing only on spatial correlations, here sensor nodes take advantage of the temporal correlations that may exist between successive sensor readings taken by a particular sensor node by using time series forecasting. Once again, cross-layer information from the MAC is used to improve sensing coverage. Our results, based on both real and synthetic data sets, indicate a reduction in sensor sampling by up to 93%, reduction in message transmissions by up to 99% and overall energy savings of up to 87%. We also show that generally more than 90% of the collected readings fall within the user-defined error threshold.

7.1.1 Reflections on the cross-layered approach

Right at the beginning of this thesis in Section 1.1.3, we hypothesized that "regardless of the class of application being addressed, a cross-layered approach would help develop solutions that are efficient, adaptive and self-organizing". We have demonstrated, with the help of several different algorithms presented in this thesis that our original hypothesis was valid.

However, one point that clearly stands out is the fact that we have only used a single MAC protocol, i.e. LMAC, to demonstrate the viability of our cross-layered approach. The basic question that comes to one's mind is whether this cross-layered approach will work with other MAC protocols.

One way of looking at this issue is to see what information is required by the higher layers to operate. For example, for the algorithms presented in this thesis, it is useful to have local topology information, time synchronization, a colouring scheme to generate independent sets, distance to gateway and reliability. Of course this list is definitely not exhaustive (e.g. certain MAC protocols could provide other information) and it is not essential to have this information provided by the MAC layer itself, i.e. there could be dedicated individual mechanisms to generate these pieces of information as well. However, for our application scenarios we have observed that having a single mechanism that not only controls the transceiver's operation but also provides us with all the information we require for the higher layers, is definitely a more efficient approach. At the time of carrying out this research, LMAC was the only MAC protocol to our knowledge that helped us pursue this cross-layered approach.

However, it is important to highlight that LMAC just like any other protocol, does have its own disadvantages. For example, a node running LMAC spends a lot of energy listening out for CM sections in every slot (except its own). However, this is the price to pay if one wishes to have a mechanism that needs

to continuously look out for changes and at the same time provides all the additional required information stated earlier. If there was any other more efficient protocol which provides the same information, we would definitely opt for it. The focus of this thesis has only been to illustrate that the frontiers of cross-layer optimization can be pushed to involve all layers above the MAC to improve the efficiency, adaptability and self-organizing capabilities of the system.

7.2 Future research directions

This thesis illustrates a number of advantages that can be derived from taking a cross-layered approach when designing sensor network applications. However, there are certain issues that need to be addressed in order to migrate these algorithms from simulations and prototypes to large-scale real world deployments. In addition, certain improvements can still be made to some of the existing protocols in order to improve performance even further. We list some of these issues and ideas below.

- *Combining spatial and temporal aggregation strategies:* We have addressed two algorithms which take advantage of spatial and temporal correlations separately. We believe the effectiveness of these algorithms would be greatly enhanced if the algorithms are combined. Thus ideally, a node should autonomously be able to decide whether to use only spatial or temporal correlations or both, depending on the characteristics of the monitored environment.

- *The need for topology stabilization mechanisms:* We have made the assumption in all the algorithms presented in this thesis that the networks are fixed, i.e. nodes in the network do not move. While nodes do remain static in fixed deployments, an issue we have not addressed is the fact that radio reception is a highly dynamic parameter even in completely static environments. For example, link quality can vary due to environmental parameters such as humidity, rain, moving objects, etc. This can have a detrimental impact on the algorithms developed in the thesis. In certain cases, it may also impact some of the bounds we have computed. In a cross-layered approach, instabilities at the MAC layer can affect the rest of the WSN protocol stack. Thus it is essential to develop filtering techniques that can help stabilize the network topology so that the higher layers are less vulnerable to disturbances at the lower layers.

- *Pushing the frontiers of cross-layer optimization even further:* In this thesis we have extended the extent of cross-layer optimization by introducing information exchange beyond the conventional MAC and routing layers to include the data management and sensor layers. This is illustrated in Figure 1.3. In the approach taken in this thesis, the sensor layer uses information provided by the MAC layer. In the future we intend to investigate if the information sensed by the sensor layer can also be used to influence the operation of the MAC layer thus making the information flow between the two layers bi-directional. Also, currently we have only considered layers above the MAC for cross-layer optimization. It would be interesting to investigate if the physical layer can also be included in the cross-layered approach.

- *Performing cross-layer optimizations based on class of application:* While a cross-layered approach can have many positive repercussions, this strategy may not be very well received by the industry. This is because it would be too expensive to work out new solutions for every new deployment. In view of this, techniques need to be developed that would help generalize cross-layer optimization strategies for a certain class of applications rather than specific applications.

- *Application-optimized MAC*: We have stated in this thesis that LMAC was found to be the most suitable MAC for carrying out cross-layer optimizations. Another approach might be to list out all the different types of information that could be gathered for a particular class of applications. This information could then be used to design the 'ideal' MAC for this specific class of applications. In other words, we could pursue the problem from a reverse direction: instead of first choosing a MAC and then carrying out optimizations based on the information provided by the MAC, it might be useful to first find out all the possible information that can be derived from a particular class of applications and then design a MAC so that it fits perfectly.

- *Include support for heterogeneous networks:* We have made the assumption throughout this thesis that all nodes in the network are able to run LMAC. This may not always be true in real life deployments. In order to reduce costs, the user may opt to have a combination of more and less powerful nodes. Only the more powerful nodes may be able to run LMAC. It would also be useful to investigate scenarios where more powerful nodes

are permanently powered. Thus solutions need to be developed to handle such heterogeneous networks.

This thesis has proposed a cross-layered approach to designing WSN protocols. We have illustrated how this may be achieved by developing a number of algorithms for various parts of the WSN protocol stack using the proposed approach. It is hoped that the strategies laid out in this thesis help us inch closer to a world where sensor networks gradually merge and disappear into the very fabric of our daily lives.

Bibliography

[1] 29 Palms Fixed/Mobile Experiment, Tracking vehicles with a UAV-delivered sensor network. http://robotics.eecs.berkeley.edu/~pister/29Palms0103/.

[2] Ambient systems, products. http://www.ambient-systems.net/ambient/products-lineup.htm.

[3] Analog devices, adxl202 low-cost 2 g dual-axis accelerometer. http://www.analog.com/en/prod/0,2877,ADXL202,00.html.

[4] BTnodes - A distributed environment for prototyping ad hoc networks. http://www.btnode.ethz.ch/.

[5] The CoBIs portal. http://www.cobis-online.de/.

[6] Crossbow, wireless sensor networks. http://www.xbow.com/Products/productdetails.aspx?sid=156.

[7] Friis transmission equation - Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Friis_transmission_equation.

[8] Hogthrob - Networked on-a-chip nodes for sow monitoring. http://www.hogthrob.dk/.

[9] Honeywell, hmc1002. http://www.ssec.honeywell.com/magnetic/datasheets/hmc1001-2&1021-2.pdf.

[10] intelliSensor: Monitoring temperatures in the zoo's penguin exhibit. http://www.sensormgmt.com/Case%20Study%20-%20Zoo.htm.

[11] Intersema, ms5534a barometer module. http://www.intersema.ch/site/technical/files/ms5534.pdf.

[12] Mars exploration rover mission. http://marsrovers.nasa.gov/home/.

[13] Melexis microelectronic integrated systems, mlx90601. http://www.dacomwest.de/pdf/mlx90601.pdf.

[14] Microbots, field and space robotics laboratory. http://robots.mit.edu/projects/microbots/index.html.

[15] Omnet++, discrete event simulator. http://www.omnetpp.org.

[16] Powercast. http://www.powercastco.com.

[17] Sensirion, sht11 sensor module. http://www.parallax.com/dl/docs/prod/acc/SensirionDocs.pdf.

[18] Texas advanced optoelectronic solutions, tsl2550 ambient light sensor. http://www.taosinc.com/product_detail.asp?cateid=4&proid=18.

[19] The TinyOS documentation project. http://ttdp.org/guides.html.

[20] USGS, The National Map Seamless Server. http://seamless.usgs.gov/.

[21] Harvesting the power of body heat. Medical Device & Diagnostic Industry, Nov 2002. http://www.devicelink.com/mddi/archive/02/11/012.html.

[22] Intel lab data. http://db.csail.mit.edu/labdata/labdata.html, 2004.

[23] RESCUENET: Embedded in-building sensor network to assist disaster rescue, 2005. http://www.rescuenet.cs.pdx.edu/index.html.

[24] Sensorware systems, 2005. http://www.sensorwaresystems.com/.

[25] Tsunami-detecting network in development, 2006. http://www.cs.pitt.edu/news/articles/2006/tsunami.php.

[26] CAALYX: Complete ambient assisted living experiment, 2007. http://caalyx.eu/.

[27] RF Monolithics, Inc, RFM TR1001 868.35mhz hybrid transceiver. http://www.rfm.com/products/data/tr1001.pdf, 2007.

[28] Coral bleaching - a sign of the times. http://www.marinephotobank.org/photocenter/stories_coralbleach.php, 2008.

[29] Falmouth scientific, inc. http://www.falmouth.com/products/index.htm, 2008.

[30] Google earth. http://earth.google.com/, 2008.

[31] D. J. Abadi, S. Madden, and W. Lindner. Reed: robust, efficient filtering and event detection in sensor networks. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 769–780. VLDB Endowment, 2005.

[32] AIMS. Reef at our fingertips. http://www.aims.gov.au/pages/about/communications/waypoint/headlines-04.html, july 2006.

[33] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks (Amsterdam, Netherlands: 1999)*, 38(4):393–422, 2002.

[34] Ambient Systems. http://www.ambient-systems.net/ambient/index.htm.

[35] S. Begum, S.-C. Wang, B. Krishnamachari, and A. Helmy. Election: Energy-efficient and low-latency scheduling technique for wireless sensor networks. In *LCN '04: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN'04)*, pages 60–67, Washington, DC, USA, 2004. IEEE Computer Society.

[36] A. Blanchard, J. Golden, R. Morgan, J. Cai, A. Fumagalli, and F. Maloberti. Megamesh sensor network design. In *Geoscience and Remote Sensing Symposium, IGARSS '03*, pages 536–538. IEEE, 2003.

[37] O. Bondarenko, S. Kininmonth, and M. Kingsford. Underwater sensor networks, oceanography and plankton assemblages. In M. Palaniswami, S. Marusic, and Y. W. Law, editors, *Proceedings of the IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing, ISSNIP 2007, Melbourne*, ISSNIP, pages 657–662, Melbourne, December 2007. IEEE.

[38] B. J. Bonfils and P. Bonnet. Adaptive and decentralized operator placement for in-network query processing. *Telecommunication Systems*, 26(2-4):389–409, 2004.

[39] P. J. Brockwell and R. A. Davis. *Introduction to Time-Series and Forecasting*. John Wiley & Sons, 1991.

[40] N. Bulusu, D. Estrin, L. Girod, and J. Heidemann. Scalable coordination for wireless sensor networks: Self-conguring localization systems. In *International Symposium on Communication Theory and Applications (ISCTA)*, Cumbria, UK, July 2001.

[41] J. Burrell, T. Brooke, and R. Beckwith. Vineyard computing: Sensor networks in agricultural production. volume 03, pages 38–45, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[42] V. Byckovskiy, S. Megerian, D. Estrin, and M. Potkonjak. A collaborative approach to in-place sensor calibration. In *Proceedings of the Second International Workshop on Information Processing in Sensor Networks (IPSN)*, volume 2634 of *Lecture Notes in Computer Science*, pages 301–316. Springer–Verlag Berlin Heidelberg, 2003.

[43] R. Cardell-Oliver, M. Kranz, K. Smettem, and K. Mayer. A reactive soil moisture sensor network: Design and field evaluation. *International Journal of Distributed Sensor Networks*, 1(2):149–162, 2005.

[44] U. Cetintemel, A. Flinders, and Y. Sun. Power-efficient data dissemination in wireless sensor networks. In *MobiDe '03: Proceedings of the 3rd ACM international workshop on Data engineering for wireless and mobile access*, pages 1–8, New York, NY, USA, 2003. ACM.

[45] S. Chatterjea and P. J. M. Havinga. A taxonomy of distributed query management techniques for wireless sensor networks. *International Journal of Communication Systems*, 20(7):889–908, January 2006.

[46] S. Chatterjea and P. J. M. Havinga. Pushing the frontiers of cross-layer optimization in wireless sensor networks right up to the application layer. In M. Palaniswami, S. Marusic, and Y. W. Law, editors, *Proceedings of the IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing, ISSNIP 2007, Melbourne*, ISSNIP, pages 19–24, Melbourne, December 2007. IEEE.

[47] S. Chatterjea, S. Kininmonth, and P. J. M. Havinga. Sensor networks. *GeoConnexion*, 5(9):20–22, October 2006.

[48] 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 TOSN*, To be published.

[49] S. Chatterjea, T. Nieberg, Y. Zhang, and P. J. M. Havinga. Energy-efficient data acquisition using a distributed and self-organizing scheduling algorithm for wireless sensor networks. In *DCOSS*, pages 368–385, 2007.

[50] S. Chatterjea, L. van Hoesel, and P. Havinga. Ai-lmac: an adaptive, information-centric and lightweight mac protocol for wireless sensor networks. *Intelligent Sensors, Sensor Networks and Information Processing Conference, 2004. Proceedings of the 2004*, pages 381–388, 14-17 Dec. 2004.

[51] D. Chu, A. Deshpande, J. M. Hellerstein, and W. Hong. Approximate data collection in sensor networks using probabilistic models. In *ICDE*, page 48, 2006.

[52] E. Cohen, B. Krishnamurthy, and J. Rexford. Efficient algorithms for predicting requests to web servers. In *INFOCOM*, pages 284–293, 1999.

[53] S. Coleri, S. Y. Cheung, and P. Varaiya. Sensor networks for monitoring traffic. In *Forty-Second Annual Allerton Conference on Communication, Control, and Computing*, Illinois, September 2004.

[54] A. Coman, J. Sander, and M. A. Nascimento. An analysis of spatio-temporal query processing in sensor networks. In *ICDEW '05: Proceedings of the 21st International Conference on Data Engineering Workshops*, page 1190, Washington, DC, USA, 2005. IEEE Computer Society.

[55] M. Conner. New battery technologies hold promise, peril for portable-system designers. EDN Magazine, May 2005. `http://www.edn.com/article/CA6288029.html`.

[56] P. Crescenzi and V. Kann. A compendium of np optimization problems: Maximum independent set. `http://www.nada.kth.se/~viggo/wwwcompendium/node34.html`, 2005.

[57] P. Crescenzi and V. Kann. A compendium of np optimization problems: Minimum independent dominating set. `http://www.nada.kth.se/~viggo/wwwcompendium/node14.html`, 2005.

[58] Crossbow. `http://www.xbow.com/Home/HomePage.aspx`.

[59] T. Dam and K. Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *First International Conference on Embedded Networked Sensor Systems*, Los Angeles, USA, 2003.

[60] A. Deshpande, C. Guestrin, S. Madden, J. M. Hellerstein, and W. Hong. Model-based approximate querying in sensor networks. *VLDB J.*, 14(4):417–443, 2005.

[61] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. 2004.

[62] E. Dijkstra. Self-stabilizing systems in spite of distributed control. *Communications of the ACM*, 17(11):643–644.

[63] S. Dolev. *Self-Stabilization*. The MIT Press, 2000.

[64] S. Dulman, S. Chatterjea, T. Hoffmeijer, P. Havinga, and J. Hurink. Architectures for wireless sensor networks. In R. Zurawski, editor, *Embedded Systems Handbook*, pages 31–1–31–10. CRC Press, Florida, 2006.

[65] S. O. Dulman. *Data-centric architecture for wireless sensor networks*. PhD thesis, Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, Enschede, The Netherlands, October 2005.

[66] J. E. Elson. *Time Synchronization in Wireless Sensor Networks*. PhD thesis, Department of Computer Science, University of California, Los Angeles, Los Angeles, USA, 2003.

[67] F. Emekci, S. E. Tuna, D. Agrawal, and A. E. Abbadi. Binocular: a system monitoring framework. In *DMSN '04: Proceedings of the 1st international workshop on Data management for sensor networks*, pages 5–9, New York, NY, USA, 2004. ACM Press.

[68] L. Evers, P. J. M. Havinga, and J. Kuper. Dynamic sensor network reprogramming using sensorscheme. In *Proceedings of the 18th Annual IEEE Symposium on Personal, Indoor and Mobile Radio Communications, Athens, Greece*, pages 1–5, Los Alamitos, September 2007. IEEE Computer Society Press.

[69] Forrester Research. Worldwide PC market to double by 2010. `http://www.forrester.com/ER/Press/Release/0,1769,971,00.html`.

[70] D. Ganesan, D. Estrin, and J. Heidemann. Dimensions: why do we need a new data handling architecture for sensor networks? *SIGCOMM Comput. Commun. Rev.*, 33(1):143–148, 2003.

[71] S. Glaser. Some real-world applications of wireless sensor nodes. In *SPIE Symposium on Smart Structures and Materials (NDE)*, San Diego, CA, USA, Mar 2004.

[72] B. Greenstein, S. Ratnasamy, S. Shenker, R. Govindan, and D. Estrin. Difs: a distributed index for features in sensor networks. *Ad Hoc Networks*, 1(2-3):333–349, 2003.

[73] V. Handziski, J. Polastre, J. Hauer, and C. Sharp. Flexible hardware abstraction of the TI MSP430 microcontroller in TinyOS. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 277–278, New York, NY, USA, 2004. ACM Press.

[74] W. R. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. 1(4):660–670, 2002.

[75] J. M. Hellerstein, W. Hong, S. Madden, and K. Stanek. Beyond average: Toward sophisticated sensing with queries. In *IPSN*, pages 63–79, 2003.

[76] T. Herman. Models of self-stabilization and sensor networks. In *IWDC*, pages 205–214, 2003.

[77] L. Hoesel. *Sensors on speaking terms : schedule-based medium access control protocols for wireless sensor networks*. PhD thesis, University of Twente, The Netherlands, 2007.

[78] L. Hoesel and P. Havinga. A lightweight medium access protocol (lmac) for wireless sensor networks: Reducing preamble transmissions and transceiver state switches. In *INSS*, Tokyo, Japan, June 2004.

[79] R. Huang and Y. Manoli. Phased array and adaptive antenna transceivers in wireless sensor networks. *dsd*, 0:587–592, 2004.

[80] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann. Impact of network density on data aggregation in wireless sensor networks. In *ICDCS '02: Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*, page 457, Washington, DC, USA, 2002. IEEE Computer Society.

[81] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 56–67, New York, NY, USA, 2000. ACM Press.

[82] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. Netw.*, 11(1):2–16, February 2003.

[83] A. Jindal and K. Psounis. Modeling spatially correlated data in sensor networks. *ACM Trans. Sen. Netw.*, 2(4):466–499, 2006.

[84] R. Jones. Managing climate change risks. 2003.

[85] B. Karp and H. T. Kung. Gpsr: greedy perimeter stateless routing for wireless networks. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254, New York, NY, USA, 2000. ACM.

[86] P. Kulkarni, D. Ganesan, and P. Shenoy. The case for multi–tier camera sensor networks. In *NOSSDAV '05: Proceedings of the international workshop on Network and operating systems support for digital audio and video*, pages 141–146, New York, NY, USA, 2005. ACM Press.

[87] K. Langendoen, A. Baggio, and O. Visser. Murphy loves potatoes: Experiences from a pilot sensor network deployment in precision agriculture. In *14th Int. Workshop on Parallel and Distributed Real-Time Systems (WPDRTS)*, pages 1–8, apr 2006.

[88] K. Langendoen and G. Halkes. Energy-efficient medium access control. *Embedded Systems Handbook, (R. Zurawski ed.) CRC Press*, 2005.

[89] Y. W. Law. *Key management and link-layer security of wireless sensor networks : Energy-efficient attack and defense*. PhD thesis, University of Twente, December 2005.

[90] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: accurate and scalable simulation of entire TinyOS applications. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 126–137, New York, NY, USA, 2003. ACM.

[91] C. Liu, K. Wu, and J. Pei. An energy efficient data collection framework for wireless sensor networks by exploiting spatiotemporal correlation. *IEEE Transactions on Parallel and Distributed Systems*, To appear.

[92] G. Lu, B. Krishnamachari, and C. S. Raghavendra. An adaptive energy-efficient and low-latency mac for data gathering in wireless sensor networks. *ipdps*, 13:224a, 2004.

[93] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, California, 1996.

[94] S. Madden. *The Design and Evaluation of a Query Processing Architecture for Sensor Networks*. PhD thesis, UC Berkeley, USA, 2003.

[95] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TinyDB: an acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.*, 30(1):122–173, 2005.

[96] S. Madden, M. Franklin, J. Hellerstein, and W.Hong. The design of an acquisitional query processor for sensor networks. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 491–502, New York, NY, USA, 2003. ACM Press.

[97] S. Madden, R. Szewczyk, M. J. Franklin, and D. Culler. Supporting aggregate queries over ad-hoc wireless sensor networks. In *WMCSA '02: Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications*, page 49, Washington, DC, USA, 2002. IEEE Computer Society.

[98] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88–97, New York, NY, USA, 2002. ACM Press.

[99] A. Manjeshwar and D. P. Agrawal. TEEN: A routing protocol for enhanced efficiency in wireless sensor networks. *IPDPS*, 03:30189a, 2001.

[100] M. Marin-Perianu and P. J. M. Havinga. Experiments with reliable data delivery in wireless sensor networks. In *Intelligent Sensors, Sensor Networks Information Processing Conference (ISSNIP)*, pages 109–114, Los Alamitos, California, December 2005. IEEE Computer Society.

[101] M. Marin-Perianu and P. J. M. Havinga. Rmd: Reliable multicast data dissemination within groups of collaborating objects. In *First IEEE International Workshop on Practical Issues in Building Sensor Network Applications, SenseApp 2006*, pages 656–663, Los Alamitos, November 2006. IEEE Computer Society Press.

[102] R. S. Marin-Perianu, M. Marin-Perianu, P. J. M. Havinga, and J. Scholten. Movement-based group awareness with wireless sensor networks. In *Pervasive 2007*, Toronto, Ontario, Canada, May 2007.

[103] Matlab. The mathworks - matlab - the language of technical computing. http://www.mathworks.com/products/matlab/, 2006.

[104] P. Mendes, A. Bartek, J. Burghartz, and J. Correia. Novel very small dual-band chip-size antenna for wireless sensor networks. *Radio and Wireless Conference, 2004 IEEE*, pages 419–422, 19-22 Sept. 2004.

[105] S. Meninger, J. Mur-Miranda, R. Amirtharajah, A. Chandrakasan, and J. Lang. Vibration-to-electric energy conversion. In *ISLPED '99: Proceedings of the 1999 international symposium on Low power electronics and design*, pages 48–53, New York, NY, USA, 1999. ACM Press.

[106] P. J. Michiel Horsman, Mihai Marin-Perianu and P. Havinga. A simulation framework for evaluating complete reprogramming solutions in wireless sensor networks. In *ISWPC 2008*, Santorini, Greece, April 2008.

[107] Microsoft Corporation. Microsoft's tradition of information. `http://www.microsoft.com/About/CompanyInformation/ourbusinesses/profile.mspx`.

[108] R. Min, M. Bhardwaj, S. Cho, A. Sinha, E. Shih, A. Wang, and A. Chandrakasan. An architecture for a power-aware distributed microsensor node, 2000.

[109] Moteiv. `http://www.moteiv.com/`.

[110] National Parks Board. Bukit timah nature reserve. `http://www.nparks.gov.sg/nature_bukit.asp`.

[111] D. Niculescu and B. Nath. Dv based positioning in ad hoc networks. *Telecommunication Systems*, 22(1-4):267–280, 2003.

[112] T. Nieberg. *Independent and dominating sets in wireless communication graphs*. PhD thesis, University of Twente, The Netherlands, 2006.

[113] C. Olston and J. Widom. Best-effort cache synchronization with source cooperation. In *SIGMOD Conference*, 2002.

[114] Omega Data Logger Home Page. `http://www.omega.com/prodinfo/dataloggers.html`.

[115] Onset - the HOBO data logger company. `http://www.1800loggers.com/`.

[116] G. Ooi, K. Chatterjea, C. Chang, and K. Lim. *Development and environment: the constant battle*, pages 145–178. Geographies of a Changing World: Global Issues in the Early 21st Century. Prentice Hall, Singapore, 2007.

[117] C. Palazzi, G. Woods, I. Atkinson, and S. Kininmonth. High speed over ocean radio link to great barrier reef. In *TENCON*, Melbourne, Australia, November 2005. IEEE.

[118] pico Technology. `http://www.picotech.com/data.html`.

[119] K. Pister. 29 palms fixed/mobile experiment. `http://robotics.eecs.berkeley.edu/~pister/29Palms0103/`.

[120] R. Colin Johnson. Energy harvesting matures. http://www.eetimes.com/news/latest/showArticle.jhtml;jsessionid=FTVGYXO23AQUOQSNDLSCKHA?articleID=204600613.

[121] J. Rabaey, E. Arens, C. Federspiel, A. Gadgil, D. Messerschmitt, W. Nazaro, K. Pister, S. Oren, and P. Varaiya. Smart energy distribution and consumption: Information technology as an enabling force, 2001.

[122] J. M. Rabaey, M. J. Ammer, J. L. da Silva, D. Patel, and S. Roundy. PicoRadio supports ad hoc ultra-low power wireless networking. *Computer*, 33(7):42–48, 2000.

[123] M. Rahimi, H. Shah, G. Sukhatme, J. Heideman, and D. Estrin. Studying the feasibility of energy harvesting in a mobile sensor network. *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, 1:19–24 vol.1, 14-19 Sept. 2003.

[124] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves. Energy-efficient, collision-free medium access control for wireless sensor networks. *Wirel. Netw.*, 12(1):63–78, 2006.

[125] N. Ramanathan, E. Kohler, and D. Estrin. Towards a debugging system for sensor networks. *Int. J. Netw. Manag.*, 15(4):223–234, 2005.

[126] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu. Data-centric storage in sensornets with ght, a geographic hash table. *Mob. Netw. Appl.*, 8(4):427–442, 2003.

[127] RFDesign. Why tiny sensors with single-chip wireless platforms are flourishing. http://rfdesign.com/mag/radio_why_tiny_sensors/.

[128] A. Sharaf, J. Beaver, A. Labrinidis, and K. Chrysanthis. Balancing energy efficiency and quality of aggregate data in sensor networks. *The VLDB Journal*, 13(4):384–403, 2004.

[129] G. Simon, M. Maroti, A. Ledeczi, G. Balogh, B. Kusy, A. Nadas, G. Pap, J. Sallai, and K. Frampton. Sensor network-based countersniper system. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 1–12, New York, NY, USA, 2004. ACM Press.

[130] V. Singhvi, A. Krause, C. Guestrin, J. Garrett, and H. Matthews. Intelligent light control using sensor networks. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 218–229, New York, NY, USA, 2005. ACM Press.

[131] A. Smyth, J. Pei, and S. Masri. System identification of the vincent thomas suspension bridge using earthquake records. *Earthquake Engineering and Structural Dynamics*, 32(3):339–367, 2003.

[132] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong. A macroscope in the redwoods. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 51–63, New York, NY, USA, 2005. ACM Press.

[133] N. Trigoni, Y. Yao, A. Demers, J. Gehrke, and R. Rajaraman. Wavescheduling: energy-efficient data dissemination for sensor networks. In *DMSN '04: Proceeedings of the 1st international workshop on Data management for sensor networks*, pages 48–57, New York, NY, USA, 2004. ACM Press.

[134] D. Tulone and S. Madden. An energy-efficient querying framework in sensor networks for detecting node similarities. In *MSWiM '06: Proceedings of the 9th ACM international symposium on Modeling analysis and simulation of wireless and mobile systems*, pages 191–300, New York, NY, USA, 2006. ACM Press.

[135] D. Tulone and S. Madden. Paq: Time series forecasting for approximate query answering in sensor networks. In *EWSN*, pages 21–37, 2006.

[136] M. Turon and J. Suh. Monitoring a mote security deployment with MOTE-VIEW. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. IEEE, 2005.

[137] L. F. W. van Hoesel, S. O. Dulman, P. J. M. Havinga, and H. J. Kip. Design of a low-power testbed for wireless sensor networks and verification. Technical Report TR-CTIT-03-45, Enschede, September 2003.

[138] L. F. W. van Hoesel and P. J. M. Havinga. Design aspects of an energy-efficient, lightweight medium access control protocol for wireless sensor networks. Technical Report TR-CTIT-06-47, Enschede, July 2006.

[139] M. C. Vuran, B. Akan, and I. F. Akyildiz. Spatio-temporal correlation: theory and applications for wireless sensor networks. *Comput. Networks*, 45(3):245–259, 2004.

[140] M. C. Vuran and I. F. Akyildiz. Spatial correlation-based collaborative medium access control in wireless sensor networks. *IEEE/ACM Transactions on Networking*, 14(2):316–329, 2006.

[141] W. W. S. Wei. *Time Series Analysis*. Addison-Wesley Publishing Company, 1990.

[142] J. Wen. A smart indoor air quality sensor network. In M. Tomizuka, C. Yun, and V. Giurgiutiu, editors, *Smart Structures and Materials 2006: Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems. Edited by Tomizuka, Masayoshi; Yun, Chung-Bang; Giurgiutiu, Victor. Proceedings of the SPIE, Volume 6174, pp. 1277-1290 (2006).*, pages 1277–1290, Apr 2006.

[143] Y. Wen, W. Zhang, R. Wolski, and N. Chohan. Simulation-based augmented reality for sensor network development. In *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 275–288, New York, NY, USA, 2007. ACM.

[144] G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz, and J. Lees. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing*, 10(2):18–25, 2006.

[145] G. Wittenburg, K. Terfloth, F. L. Villafuerte, T. Naumowicz, H. Ritter, and J. Schiller. Fence Monitoring - Experimental Evaluation of a Use Case for Wireless Sensor Networks. In *Proceedings of the 4th European Conference on Wireless Sensor Networks (EWSN'07)*, Delft, The Netherlands, Jan. 2007.

[146] A. Woo and D. E. Culler. A transmission control scheme for media access in sensor networks. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 221–235, New York, NY, USA, 2001. ACM.

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

[148] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In *INFOCOM*, New York, USA, June 2002.

[149] Y. Yu, D. Estrin, M. Rahimi, and R. Govindan. Using more realistic data models to evaluate sensor network data processing algorithms. *lcn*, 00:569–570, 2004.

[150] V. I. Zadorozhny, P. K. Chrysanthis, and P. Krishnamurthy. A framework for extending the synergy between mac layer and query optimization in sensor networks. In *DMSN '04: Proceeedings of the 1st international workshop on Data management for sensor networks*, pages 68–77, New York, NY, USA, 2004. ACM.

[151] Y. Zhang, S. Chatterjea, and P. J. M. Havinga. Experiences with implementing a distributed and self-organizing scheduling algorithm for energy-efficient data gathering on a real-life sensor network platform. In *WOWMOM*, pages 1–5, 2007.

[152] Y. Zhang, N. Meratnia, and P. Havinga. Why general outlier detection techniques do not suffice for wireless sensor networks? *Intelligent Techniques for Warehousing and Mining Sensor Network Data*, 2008.

[153] H. Zimmermann. OSI reference model the ISO model of architecture for open systems interconnection. pages 2–9, Norwood, MA, USA, 1988. Artech House, Inc.

Publications

Journals

1. 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. In *ACM Transactions on Sensor Networks*, To appear.

2. S. Chatterjea and P. Havinga. A Taxonomy of Distributed Query Management Techniques for Wireless Sensor Networks. In *International Journal of Communication Systems*, 20 (7). pp. 889-908, Wiley, 2006.

3. S. Chatterjea, L.F.W. van Hoesel and P. Havinga. A Framework for a Distributed and Adaptive Query Processing Engine for Wireless Sensor Networks. In *Transactions of the Society of Instrument and Control Engineers*, E-S-1 (1). pp. 58-67, 2006.

Book chapters

4. S. Dulman, S. Chatterjea, T. Hoffmeijer, P. Havinga, and J. Hurink. Introduction to Wireless Sensor Networks. In *Embedded Systems Handbook, (R. Zurawski ed.)*, CRC Press, August 2005.

5. S. Dulman, S. Chatterjea, T. Hoffmeijer, P. Havinga, and J. Hurink. Architectures for Wireless Sensor Networks. In *Embedded Systems Handbook, (R. Zurawski ed.)*, CRC Press, August 2005.

Conferences and workshops

6. S. Chatterjea, and P. Havinga. An Adaptive and Autonomous Sensor Sampling Frequency Control Scheme for Energy-Efficient Data Acquisition in Wireless Sensor Networks. In *Proceedings of the Fourth IEEE Conference on Distributed Computing in Sensor Systems* (DCOSS), June 2008, Santorini, Greece. Pages to appear. Lecture Notes in Computer Science. Springer Verlag.

7. S. Chatterjea, and P. Havinga. Pushing the Frontiers of Cross-layer Optimization in Wireless Sensor Networks Right up to the Application Layer. In *Proceedings of the Third IEEE Conference on Intelligent Sensors, Sensor Networks and Information Processing* (ISSNIP), December 2007, Melbourne, Australia, IEEE Computer Society Press.

8. S. Chatterjea, T. Nieberg, Y. Zhang and P. Havinga. Energy-Efficient Data Acquisition using a Distributed and Self-organizing Scheduling Algorithm for Wireless Sensor Networks. In *Proceedings of the Third IEEE Conference on Distributed Computing in Sensor Systems* (DCOSS), June 2007, Santa Fe, USA. pp. 368-385. Lecture Notes in Computer Science 4549 (LNCS4549). Springer Verlag.

9. Y. Zhang, S. Chatterjea and P. Havinga. Experiences with Implementing a Distributed and Self-Organizing Scheduling Algorithm for Energy-Efficient Data Gathering on a Real-Life Sensor Network Platform. In *Proceedings of the IEEE International Symposium of World of Wireless, Mobile and Multimedia Networks* (WoWMoM), June 2007, Helsinki, Finland. pp.1-5, IEEE Computer Society Press.

10. S. Chatterjea, S. de Luigi and P. Havinga. An Adaptive, Directed Query Dissemination Scheme for Wireless Sensor Networks. In *Proceedings of the Thirty-Fifth IEEE Conference on Parallel Processing Workshops* (ICPPW), August 2006, Columbus, USA. pp. 181-188, IEEE Computer Society Press.

11. S. Chatterjea, S. de Luigi and P. Havinga. DirQ: A Directed Query Dissemination Scheme for Wireless Sensor Networks. In *Proceedings of the IASTED Conference on Wireless Sensor Networks* (WSN), July 2006, Banff, Canada. ACTA Press.

12. S. Chatterjea, L.F.W. van Hoesel and P. Havinga. An Application-Tailored MAC Protocol for Wireless Sensor Networks. In *Proceedings of the First*

International Workshop on Wireless Ad-hoc Networks (IWWAN), May 2005, London, UK.

13. S. Chatterjea, L.F.W. van Hoesel and P. Havinga. AI-LMAC: An adaptive, information-centric and lightweight MAC protocol for wireless sensor networks. In *Proceedings of the First IEEE Conference on Intelligent Sensors, Sensor Networks and Information Processing* (ISSNIP), December 2004, Melbourne, Australia, IEEE Computer Society Press.

14. L.F.W. van Hoesel, S. Chatterjea and P. Havinga. A Low-Latency, Information-Centric Medium Access Protocol for Wireless Sensor Networks. In *Proceedings of the Program for Research on Integrated Systems and Circuits* (ProRISC), November 2004, Veldhoven, The Netherlands.

15. S. Chatterjea and P. Havinga. A Framework for a Distributed and Adaptive Query Processing Engine for Wireless Sensor Networks. In *Proceedings of the First International Conference on Networked Sensing Systems* (INSS 2004), June 2004, Tokyo, Japan.

16. S. Chatterjea and P. Havinga. A Distributed Query Processing Engine. In *Proceedings of the 37th ESLAB Symposium - Tools and Technologies for Future Planetary Exploration*, European Space Research and Technology Centre, December 2003, Noordwijk, The Netherlands.

17. S. Chatterjea and P. Havinga. A Dynamic Data Aggregation Scheme for Wireless Sensor Networks. In *Proceedings of the Program for Research on Integrated Systems and Circuits* (ProRISC), November 2003, Veldhoven, The Netherlands.

18. L.F.W. van Hoesel, S. Chatterjea and P. Havinga. An Energy Efficient Medium Access Protocol for Wireless Sensor Networks. In *Proceedings of the Program for Research on Integrated Systems and Circuits* (ProRISC), November 2003, Veldhoven, The Netherlands.

19. S. Chatterjea and P. Havinga. CLUDDA - Clustered Diffusion with Dynamic Data Aggregation. In *Proceedings of the 8th Cabernet Radicals Workshop*, October 2003, Corsica, France.

Technical Reports

20. 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. TR-CTIT-07-10, 2007.

Wireless sensor networks (WSNs) have the potential to change the way we view our world today. Unlike conventional environmental monitoring techniques which depend on point measurements, large and dense WSN deployments would enable users to observe the environment at fine-grained spatial resolutions. However, the fact that it is impossible to configure and maintain such large scale networks manually, makes it essential for any WSN to have inherent self-organizing capabilities.

Extracting vast amounts of data from large scale WSNs, however, typically results in a myriad of problems ranging from excessive power consumption to high packet loss due to bottlenecks in the network.

This thesis addresses these problems by presenting several solutions for two classes of applications: (i) applications that inject queries into the network to extract only a subset of all the available data and (ii) applications that require all the data to be extracted from every sensor node in the network at periodic intervals. The advantage of using a cross-layered approach in order to incorporate self-organizing characteristics is demonstrated in every instance.

CTIT
Centre for Telematics and Information Technology

University of Twente
The Netherlands

ISBN 978090-365-2721-7