Reliable Routing Protocols for Dynamic Wireless Ad Hoc and Sensor Networks

Jian Wu

Composition of the Graduation Committee:

Prof. Dr. Ir.	A.J. Mouthaan	University of Twente, chairman and secretary
Prof. Dr. Ir.	Th. Krol	University of Twente, promotor
Dr. Ing.	P.G.M. Havinga	University of Twente, assistant promotor
Prof. Dr. Ir.	G.J.M. Smit	University of Twente
Prof. Dr. Ir.	C. Slump	University of Twente
Prof. Dr. Ir.	I.G.M.M. Niemegeers	Delft University of Technology
Dr.	J.L. Hurink	University of Twente
Dr.	J.J. Lukkien	Eindhoven University of Technology

Printed by Ipskamp PrintPartners, Enschede, the Netherlands.
Cover designed by Yu Wang.

ISBN 978-90-365-2596-1

RELIABLE ROUTING PROTOCOLS FOR DYNAMIC WIRELESS AD HOC AND SENSOR NETWORKS

DISSERTATION

to obtain
the degree of doctor at the University of Twente,
on the authority of the rector magnificus,
prof.dr. W.H.M. Zijm,
on account of the decision of the graduation committee,
to be publicly defended
on Wednesday, December 19th, 2007, at 13.15

by

Jian Wu

born on March 14th, 1977
in Beijing, China

This dissertation is approved by:

Prof. Dr. Ir.	Thijs Krol	(promotor)
Dr. Ing.	Paul Havinga	(assistant promotor)

To my wife Yu

Acknowledgement

The thesis is the completion of my PhD study at the University of Twente, where I benefit a lot from the inspiring and friendly atmospheres. I am very grateful to many people who contributed directly or indirectly to this thesis. This thesis would not have been possible without the support of them. I would like to take this opportunity to thank all of them, including those whose names are not mentioned below.

First and foremost I would like to express my gratitude to my promoter Prof. Thijs Krol and assistant promoter Dr. Paul Havinga, who gave me the opportunity to do this Ph.D research. Thijs gave me the freedom to try out new ideas and gave me continuous support during the research. I particularly appreciate Paul for his great guidance and continues support. He also gave many useful suggestions and comments on the thesis. I particularly appreciate his great patience in teaching me, many instructive discussions and fruitful work. In a word, the combination of both supervisors substantially contributed to my successful research work.

I am also grateful to the members of the graduation committee for their willingness to participate in my graduation committee. Especially thank Prof. Gerard Smit and Dr. Johann Hurink, who gave useful comments on the earlier version of the thesis.

Further, I would like to thank my roommates and colleagues: Stefan Dulman, Tim Nieberg, Lodewijk van Hoesel, Supriyo Chatterjea and Tjerk Hofmeijer. It is my pleasure to work with them and thanks for all the helps from them. I still miss the dart games in our office, although sometimes it became a little bit "dangerous". It was a memorable time to stay with them.

Moreover I would like to thank all my Chinese friends in Enschede. With these friends I have a wonderful life in the city. So much fun and good times we have shared together. The names are simply too many to mention all of them in here, and even then I would regret if I miss anyone.

Thanks to my parents, brother and sister, who always support my decision to continue my study, thanks for their unconditional support. Last but not the least,

I would like to express my deepest gratitude to my wife, Wang Yu, for her love, patient and fully support.

Jian Wu
November, 2007
Enschede, The Netherlands

Abstract

The vision of ubiquitous computing requires the development of devices and technologies, which can be pervasive without being intrusive. The basic components of such a smart environment will be small nodes with sensing and wireless communications capabilities, able to organize flexibly into a network for data collection and delivery. The constant improvements in digital circuit technology, has made the deployment of such small, inexpensive, low-power, distributed devices, which are capable of information gathering, processing, and communication in miniature packaging, a reality.

Realizing such a network presents very significant challenges, especially at the protocol and software level. Major steps forward are required in the field of communications protocol, data processing, and application support. Although sensor nodes will be equipped with a power supply (battery) and embedded processor that makes them autonomous and self-aware, their functionality and capabilities will be very limited. The resource limitations of Wireless Sensor Networks (WSN), especially in terms of energy, require novel and collaborative approach for the wireless communication. Therefore, collaboration between nodes is essential to deliver smart services in a ubiquitous setting. Current research in this area generally assumes a rather static network, leading to a strong performance degradation in a dynamic environment. In this thesis we investigate new algorithms for routing in dynamic wireless environment and evaluate their feasibility through experimentation. These algorithms will be key for building self-organizing and collaborative sensor networks that show emergent behavior and can operate in a challenging environment where nodes move, fail and energy is a scarce resource.

We develop the technology needed for building self-organizing and collaborative sensor networks using reconfigurable smart sensor nodes, which are self-aware, self-reconfigurable and autonomous. This technology will enable the creation of a new generation of sensors, which can effectively network together so as to provide a flexible platform for the support of a large variety of mobile sensor network applications. In this thesis, we address the dynamics of sink nodes, sensor nodes and event in the routing of wireless sensor networks, while maintaining high reliability

and low energy consumption. The hypothesis is that this requires different routing protocols and approaches. The varying application scenarios of wireless sensor networks require different routing protocols and approaches as well.

This thesis has three major contributions to the routing in dynamic wireless sensor networks. Firstly, a combination between a new multipath on-Demand Routing protocol and a data-splitting scheme which results in an efficient solution for high reliability and low traffic. Secondly, a cross-layered approach with a self-organizing medium access control protocol and a tightly integrated source routing protocol is designed for high mobility sensor networks. Finally, a data-centric approach based on cost estimation is designed to disseminate aggregated data from data source to destination with high efficiency.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

From 1991 on Mark Weiser published what are considered some of the seminal papers in *Ubiquitous Computing* [104] [105], which envisioned a future where everything – from doorknobs to toilet paper holders, are intelligent and connected. Ubiquitous computing (or pervasive computing) integrates computing in the environment, rather than having computers as distinct objects. Currently, the art of ubiquitous computing is not as mature as Weiser had hoped, but a considerable amount of development is currently taking place. Wireless Sensor Network (WSN) is one of the enabling technologies.

With constant improvements in digital circuit technology, the processing and storage capacity of a cost-effective chip doubles every 18 months, according to Moore's law. While it has provided ever more computing power, researchers are now applying this technology in ways that enable a new role for computing in science. A given computing capacity becomes exponentially smaller and cheaper with each passing year. Researchers can use the semiconductor manufacturing techniques that underlie this miniaturization to build radios and exceptionally small mechanical structures that sense physical conditions and chemical compounds in the surrounding environment. This technological improvement has made the deployment of small, inexpensive, low-power, distributed devices, which are capable of information gathering, processing, and communication in miniature packaging, a reality. Such devices are called sensor nodes.

Information gathering functionalities are the actual sensors, e.g., devices used to measure temperature, air pollution, pressure, etc. Information processing functionalities consist essentially of processing units like microprocessors or microcontrollers, memory and their interconnection. Information communication functionalities are the wireless communication device, which typically communicates in the radio spectrum, sometimes also in visible or infrared light. Finally, some supporting

functionalities such as power supplies, are required to build a complete sensor node.

Each sensor node is capable of only a limited amount of processing. However, when coordinated with the information from a large number of other nodes, they have the ability to measure a given physical phenomena in great detail. Thus, a sensor network can be described as a collection of sensor nodes which coordinate to perform some specific action. Unlike traditional networks, sensor networks depend on dense deployment and coordination to carry out their tasks. They differ considerably from current networked and embedded systems. They combine the large scale and distributed nature of networked systems, such as the Internet, with the extreme energy constraints and physically coupled nature of embedded control systems.

1.1 Wireless sensor networks

In the wireless sensor network, the devices need to communicate with other devices in an ad hoc fashion, without the help of an external infrastructure. Moreover, the communication technology must be robust, scale well, and use the limited energy of the device efficiently. So far, no single communication technology has established itself in the field: many current wireless communication technologies seem to lack robustness, consume too much energy, or require an external infrastructure to be viable candidates.

Routing in wireless sensor networks is very challenging due to the characteristics that distinguish them from tradition wireless networks. In this thesis, we address in particular the reliability and energy efficiency of routing in a dynamic wireless sensor network environment.

1.1.1 Applications of WSNs

Wireless sensor networks (WSN) are one of the first real world examples of pervasive computing, the notion that small, smart, and cheap sensing and computing devices will eventually permeate the environment. Though the technology is still in its early phase, it will enable a plethora of new services and applications [4].

Environmental applications: Many initial WSNs have been deployed for environmental monitoring, which involves tracking the movements of small animals and monitoring environmental conditions that affect crops and livestock. In these applications, WSNs collect readings over time across a space large enough to exhibit significant internal variation. Other applications of WSNs are chemical and biological detection, precision agriculture, biological, forest fire detection, meteorological or geophysical research, flood detection and pollution study. The College of the

Atlantic deployed a WSN consisting of 32 nodes on Great Duck Island, off the coast of Maine, to monitor Leach's Storm Petrel without human disturbance [57]. More recently, researchers from Australia, the Netherlands and the USA are developing a distributed environmental monitoring sensor network in Australia's Great Barrier Reef [47], which will provide valuable scientific information enabling a greater understanding of the environmental impact due to pollution, urban development and climate change.

Healthcare applications: The wireless sensor networks could provide interfaces for disabled, integrated patient monitoring. It can monitor and detect elderly people's behavior, e.g., when a patient has fallen. These small sensor nodes allow patients a greater freedom of movement and allow doctors to identify pre-defined symptoms earlier on [68]. The small installed sensor can also enable tracking and monitoring of doctors and patients inside a hospital. Each patient has small and lightweight sensor nodes attached to them, which may be detecting the heart rate and blood pressure. Doctors may also carry a sensor node, which allows other doctors to locate them within the hospital. MoteTrack [56] is the patient tracking system developed by Harvard University, which tracks the location of individual patients devices indoors and outdoors, using radio signal information from the sensor attached to the patients.

Home applications: With the advance of technology, the tiny sensor nodes can be embedded into furniture and appliances, such as vacuum cleaners, microwave ovens and refrigerators. They are able to communicate with each other and the room server to learn about the services they offer, e.g., printing, scanning and faxing. These room servers and sensor nodes can be integrated with existing embedded devices to become self-organizing, self-regulated and adaptive systems to form a smart environment. One example of a smart environment is the Residential Laboratory at Georgia Institute of Technology [24].

Military applications: The rapid deployment, self-organization and fault tolerance characteristics of sensor networks make them a very promising sensing technique for military applications. Military sensor networks could be used to detect and gain as much information as possible about enemy movements, explosions, and other phenomena of interest, such as battlefield surveillance, nuclear, biological and chemical attack detection and reconnaissance. As an example, sensors help to locate the source of incoming small arms fire so that counterattacks can be launched against snipers quickly and precisely [90].

Other commercial applications: The advance of wireless sensor networks leads to many agricultural, industrial and commercial applications. Some examples are herd monitoring, building virtual keyboards, managing inventory, monitoring product quality, interactive toys and transportation management.

1.1.2 Wireless sensor node architecture

Sensor networks are expected to be left unattended for a long period of time. Each sensor is running on batteries, which requires an approach that explicitly takes energy into consideration. For this, each node is made aware of its energy requirements and usage by a model of the energy consumption. Various types of sensor nodes have been developed for academic research [79] [13] [99] [12] [117], as well as for the market [2] [48] [65]. The system architecture of a generic wireless sensor node is comprised of four subsystems [82]:

- a computing subsystem consisting of a microprocessor,
- a communication subsystem consisting of a short range radio for wireless communication,
- a sensing subsystem that links the node to the physical world and consists of a group of sensors and actuators, and
- a power supply subsystem, which houses the battery and the (optional) DC-DC converter, and powers the rest of the node.

The sensor node shown in Figure 1.1 on page 4 is representative for commonly used node architectures. Each subsystem plays an important role in the sensor node.

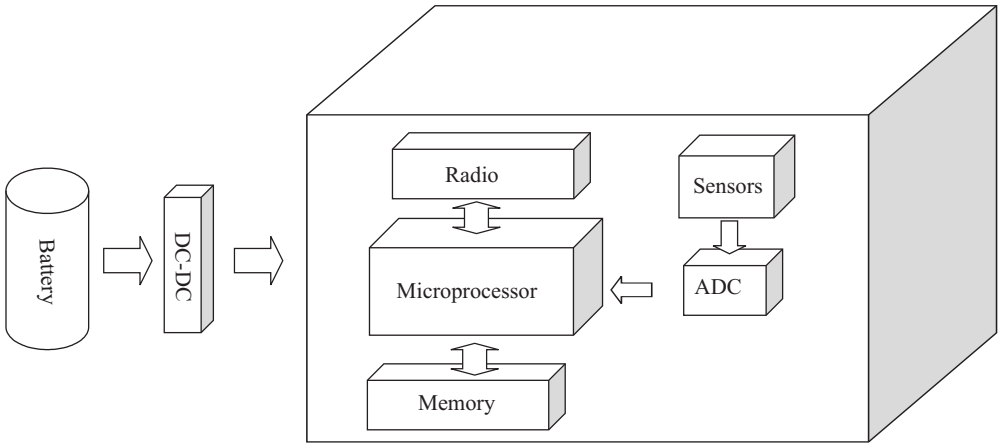


Figure 1.1: System architecture of a typical wireless sensor node

- *Radio* - Enabling wireless communication with neighboring nodes and the outside world. It consists of a short range radio which usually has a single channel, a low data rate and operates at unlicensed bands of 868-870 MHz

(EU), 902-928 MHz (Americas) or near 2.4 GHz (global), such as the most popular radios from Chipcon and RFM. There are several factors that affect the power consumption characteristics of a radio, including the type of modulation scheme used, data rate, transmit power (determined by the transmission distance) and the operational duty cycle. For the energy consumption, four different states need to be distinguished: *Transmit, Receive, Idle* and *Standby* modes. An important observation in the case of most radios is that, operating in *Idle* mode results in significantly high power consumption, almost equal to the power consumed in the *Receive* mode [98]. Thus, it is important to completely shut down the radio rather than set it in the *Idle* mode when it is not transmitting or receiving due to the high power consumed. Another influencing factor is that, as the radio's operating mode changes, the transient activity in the radio electronics causes a significant amount of power dissipation. For example, when the radio switches from *standby* mode to *Transmit* mode to send a packet, a significant amount of power is consumed for starting up the transmitter itself [102].

- *Microprocessor* - Providing intelligence to the sensor node. The Microprocessor is responsible for control of the sensors, and execution of communication protocols and signal processing algorithms on the gathered sensor data. Commonly used microprocessors are Intel's StrongARM microprocessor, Atmel's AVR microcontroller and Texas Instruments' MP430 microprocessor. In general, four main processor states can be identified in a microprocessor: *off, sleep, idle,* and *active*. In sleep mode, the CPU and most internal peripherals are turned off, and can only be activated by an external event (interrupt). In idle mode, the CPU is still inactive, but other peripherals are active, for example, the internal clock or timer. In the active state, multiple substates may be defined based on clock speeds and voltages. Within the active states, the CPU and all peripherals are active. The power-performance characteristics of MCUs have been studied extensively and several techniques have been proposed to estimate the power consumption of these embedded processors [94] [72].

- *Sensor* - Translating physical phenomena to electrical signals. Sensors can be classified as either analog or digital devices. There exists a variety of sensors that measure environmental parameters such as temperature, light intensity, sound, magnetic fields, image, etc. Given the diversity of sensors, there is no standard power consumption figure. For a simple sensor we assume that only the states *on* and *off* are given, and that the energy consumption within both states can be measured by time. However, more powerful sensors operate in different states, comparable to the microprocessor. Energy consumption can be reduced by using low power components and saving power at the cost of performance which is not required.

- *Battery* - The battery supplies power to the complete sensor node, and hence plays a vital role in determining sensor node lifetime. The amount of power drawn from a battery should be checked. As sensor nodes are generally small, light and cheap, the size of the battery is limited. Battery technologies advance much more slowly than semiconductor technologies. For example, the energy densities of Li-ion batteries only increased 50% from 1999 to 1994 [8]. While in the same period of time, the number of transistors of Intel processors doubles every 24 months [37]. Furthermore, sensors must have a lifetime of months to years, since battery replacement is not an option for networks with thousands of physically embedded nodes. This causes energy consumption to be the most important factor in determining sensor node lifetime.

1.1.3 Routing in wireless sensor networks

The basic purpose of a wireless sensor network is to monitor the physical environment, and provide this information in an appropriate fashion to applications when needed. Each node is equipped with one or more sensors, whose readings are transported via other network nodes to a data sink. This section gives an introduction on the communication types, operation modes and communication patterns of wireless sensor networks.

In general, two types of nodes are logically recognized: nodes that mainly sense the physical data and transmit their own sensor readings (*sensor nodes*) and nodes that mainly relay messages from other nodes (*relay nodes*). Sensor readings are routed from the source nodes to the sink via the relay nodes, thus creating a multi-hop topology. This logical organization implies four types of communications types as shown in Figure 1.2 on page 6, that must be accounted for, especially on lower communication levels such as the MAC protocol.

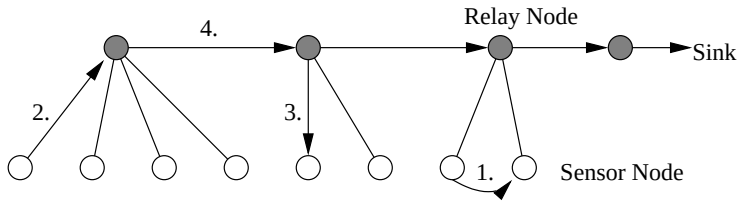


Figure 1.2: WSN Communication Types

1. *Sensor node to sensor node communication* - This direct type of communication is used for local operations, for example, during the clustering process, or the route creation process.

2. *Sensor node to relay node communication* - Sensor data is transmitted from a sensor node to a relay node. This type of communication is often unicast.

3. *Relay node to sensor node communication* - Requests for data and signaling messages (often multicasts) to reach a subset of the surrounding nodes at once, are spread by the relay nodes.

4. *Relay node to relay node communication* - The relay nodes form the backbone of the network. Communication between these nodes is mostly unicast. Note that every node is equipped with a wireless transceiver and thus is able to perform the duties of a relay node.

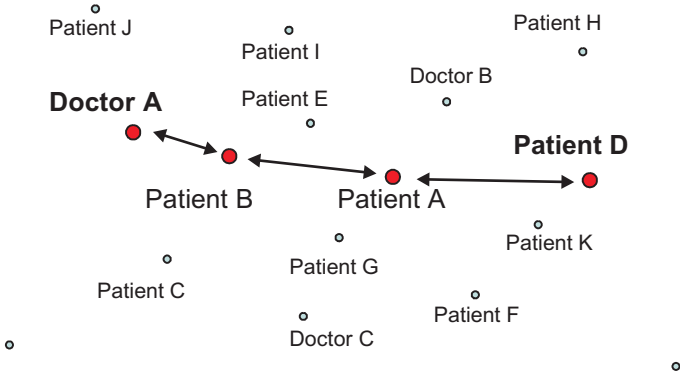


Figure 1.3: The point-point communication between Doctor A and Patient D

For a sensor node, there are two operational modes to fit the possible applications arising from the WSN: 1) active polling and 2) passive detection and notification. To get the reading of a sensor, the node acting as sink can actively ask for the information from a specific node or a group of nodes (*active polling*), or request to be notified when an event is detected by one of the nodes, e.g., if a pre-determined threshold on a sensor reading is passed (*passive notification*).

On the routing level, two types of communication patterns exist between data sink and data source in the wireless sensor network, which are:

1. *Point-point communication* - In the point-point communication, the data sink is only interested in sensing the data from an individual sensor node and needs to communicate with this particular sensor via a routing protocol. For example, in a patient tracking application, the doctor needs to know the monitored data from a specific patient, who is attached with a sensor node and might

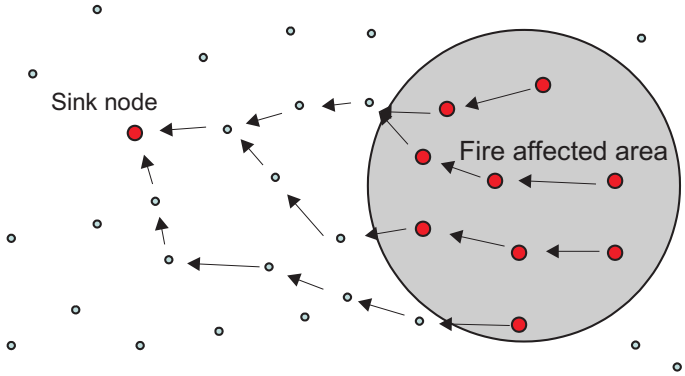


Figure 1.4: The point-multipoint communication in a forest fire detection application

move around inside the hospital. A point-point communication should be setup between the doctor node and the patient node to track the status of the individual patient as shown in Figure 1.3 on page 7.

2. *Point-multipoint communication* - In the point-multipoint communication, data sink cares about a certain type of data. Multiple sensor nodes might process this type of data and all of them send their data to the sink. A typical application scenario is found in forest fire detection, where the administration node needs to know the status from all the fire affected areas. This involves a large amount of sensors. All of these sensors should send their data through point-multipoint communication to the sink node as shown in Figure 1.4 on page 8.

In Table 1.1 on page 9, we identify some of the typical wireless sensor applications and their corresponding communication patterns.

1.2 Problem statement

Notwithstanding many similarities between wireless sensor networks and wireless ad hoc networks, sensor networks have their own unique characteristics which create new challenges for the design of wireless sensor networks. Routing in WSNs is very challenging due to the inherent characteristics that distinguish them from other wireless networks. To illustrate this point, the differences between sensor networks and existing ad hoc network concepts [95] are outlined below:

Table 1.1: Typical wireless sensor network application and its communication patterns

Application	Category	Mobility	Comm. patterns	Examples
Habitat Monitoring	environmental application	static	point-multipoint	Great Duck Island [57], Great Barrier Reef [47]
Glacier Monitoring	environmental application	static	point-multipoint	GLACSWEB at Briksdalsbreen [62]
Patient Tracking	healthcare application	mobile	point-point	MoteTrack [56]
Smart home	home application	(partly) mobile	point-point	Residential Laboratory [24]
Intrusion detection	military application	mobile	point-multipoint	Sniper localization [90]
Battle field Surveillance	military application	static	point-multipoint	Self-Healing Mine Field [63]
Herd monitoring	commercial application	mobile	point-multipoint	Networked Cows [63]
Avalanche victim rescue	commercial application	mobile	point-point	Wearable Sensor for Avalanche Rescue [64]
Power Monitoring	commercial application	static	point-point	Office Power Mointoring [46]

- *Scalability* - The number of nodes in a sensor network is significantly larger than the amount of nodes in a typical wireless ad hoc network. The difference can be several orders of magnitude, which requires different, more scalable solutions.

- *Fault tolerance* - Sensor nodes are prone to failures. Some sensor nodes may fail or be blocked due to lack of power, physical damage or environmental interference. The probability of failure in WSN is much higher than in ad hoc networks. The failure of a few sensor nodes should not affect the overall task of the sensor network. Fault tolerance is the ability to sustain sensor network functionalities without any interruption from sensor node failures [38] .

- *Energy* - Sensor nodes are not connected to any wired energy source. Often,

the battery of a sensor node is not rechargeable and the need to prolong the lifetime of a sensor node has a deep impact on the system and networking architecture. There is only a finite source of energy, which must be optimally used for processing and communication. Hence, energy consumption is a primary metric to be considered. Unlike traditional networks, where the focus is on maximizing throughput or minimizing node deployment, the major consideration in a sensor network is to extend the system lifetime, as well as extend system robustness. An interesting fact is that communication dominates processing in energy consumption. Thus, in order to make optimal use of energy, communication should be minimized as much as possible.

- *Simplicity* - Sensors are usually low-cost devices with severe constraints in respect to energy source, power, computational capability and memory. Thus the operating and networking software of a sensor node must be kept simpler as compared to desktop computers. This simplicity sometimes requires a break with conventional layering rules for networking software, since abstractions typically cost time and space.

- *Redundancy* - Sensor nodes are densely deployed. The low cost and low energy supply in many application scenarios requires redundant deployment of wireless sensor nodes. As a consequence, the importance of any one particular node is considerably reduced when compared with traditional networks. This densely deployed network enables both data aggregation and multipath routing. Data aggregation is a method of accumulating data into one aggreagted record that is otherwise presented through multiple records.

- *Dynamic changes* - The topology of a sensor network changes very frequently due to mobility and failures. A sensor network system must be adaptable to changing connectivity (e.g., due to addition of more nodes, failure of nodes, etc.), as well as changing environmental stimuli.

- *Ad hoc deployment* - Sensor nodes often use broadcast communication paradigms whereas most ad hoc networks are based on point-to-point communications. Most sensor nodes are deployed in regions which have no infrastructure at all. A typical method to deploy a wireless sensor network in a forest is to throw the sensor nodes from an aeroplane. In this situation, it is up to the nodes to identify connectivity and distribution.

- *Application specific* - Due to the large number of conceivable combinations of sensing, computing and communication technologies, many different application scenarios for wireless sensor network are possible as stated in the previous section. It is unlikely that there will be a one-thing-fits-all solution for every different possibility.

Taking into account these characteristic of wireless sensor networks, we focus this thesis on the reliability and energy efficiency of routing protocol for dynamic wireless sensor networks.

Most of the current routing protocols for WSN assume that sensor nodes are stationary. However, mobility of both data sinks and sensor nodes is inevitable in many applications, such as herd monitoring, transportation management and personal tracking. Routing messages from or to moving nodes is more challenging since route stability becomes an important issue. Current solutions in routing protocols spend an excessive amount of energy to stabilize the network under mobility. Even in a stationary network, the topology of sensor networks can still change due to node failures. Moreover, the sensed phenomenon can be either dynamic or static depending on the application, e.g., it is dynamic in a target detection/tracking application and static in forest monitoring for early fire prevention. Monitoring static events allows the network to work in a reactive mode, simply generating traffic when reporting. Dynamic events in most applications require periodic reporting and consequently generate significant traffic routed to the data sink.

In this thesis, we address the dynamics of sink nodes, sensor nodes and event in the routing of wireless sensor networks, while maintaining high reliability and low energy consumption. The hypothesis is that this requires different routing protocols and approaches. The varying application scenarios of wireless sensor networks require different routing protocols and approaches as well.

1.2.1 Cross-layer approach

Our lessons learned in developing network protocols for wireless sensor networks in the last couple of years, show that the traditional layered networking approach has several drawbacks in the resulting performance and efficiency of the system. Quite often, significant improvements are possible for the network protocols; yet they require a large amount of information to be passed along the layers of the system. In principle this approach allows independency between the various protocols, however, it incurs a significant overhead in parameter transfer. Moreover, improvements performed in a specific layer can have impairments and even be counterproductive for other layers.

Optimization is more effective when taking the overall system into account and use all available knowledge. When this information is distributed to other sensor nodes, the overhead effect is even larger. A solution in which such information is piggy-backed to other messages, can limit this extra message exchange. During the development of various protocols and services (like localization protocols), the lowest layers of our system (e.g., the MAC layer) were increasingly used to pass

information to the higher layers. The overall result of these developments has led to the cross-layered approach as described in Chapter 4 of this thesis.

1.3 Contributions and thesis outline

1.3.1 Contributions

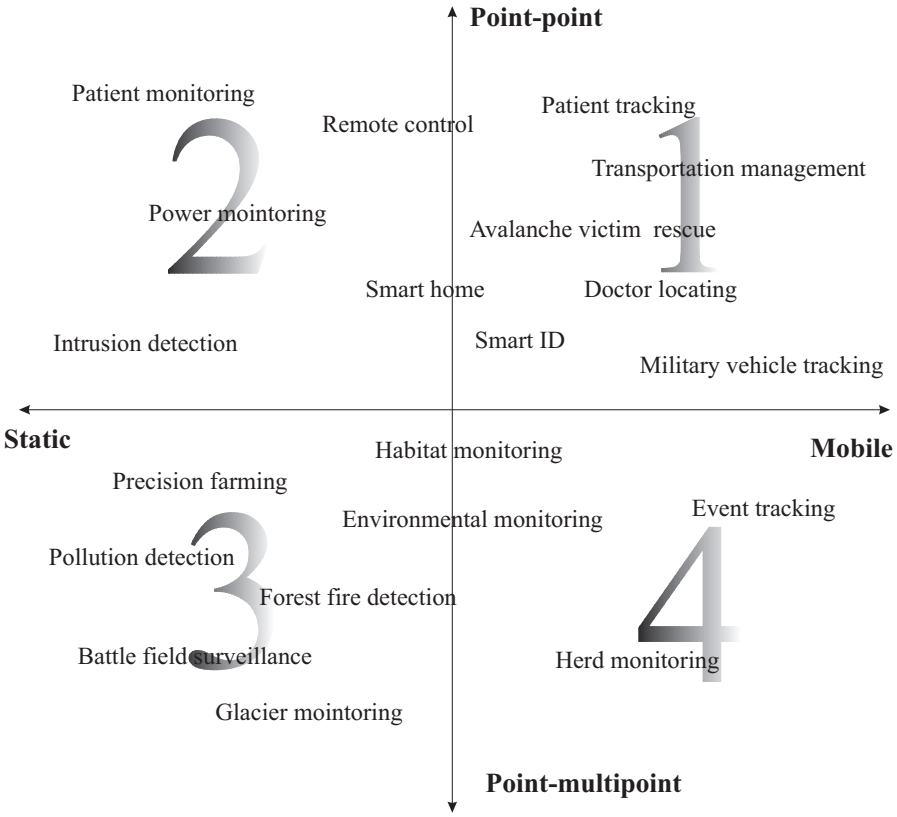


Figure 1.5: The dynamics and communication patterns of typical WSN applications

As shown in Figure 1.5 on page 12, we have located a number of typical WSN applications in the graph according to their dynamics and communication patterns. This thesis has three major contributions to the routing in dynamic wireless sensor networks.

- *Multipath on-Demand Routing (MDR)*: The first contribution lies in the first quadrant of Figure 1.5 on page 12. It addresses the reliability of dynamic wireless sensor networks in the point-point routing scenarios by multipath

routing. Some of the applications of WSN, like streaming or code dissemination, requires distributing a large amount of bulk data to the destination with high reliability and low delay. Taking into account these applications, we analyze the combination between a new Multipath on-Demand Routing (MDR) protocol and a data-splitting scheme that results in an efficient solution for achieving high delivery ratios while keeping the traffic at a low value. The algorithms presented assure that the gathered data reaches its destination in the network by assuming that some nodes may not be available during the routing procedure. Additional energy will be required for only a small amount of computations; this is almost negligible compared with the energy used for communications.

- *EYES Source Routing (ESR)*: The second contributions also lies in the first quadrant of Figure 1.5 on page 12. It addresses the reliability of dynamic wireless sensor networks in the point-point routing scenarios by cross-layer interaction. In WSN, the traditional layered networking approach in developing network protocols has several drawbacks in the resulting performance and efficiency of the system. This is especially the case in the application area of high mobility, such as vehicle tracking and personal tracking. This contribution addresses a cross-layered approach for networking in wireless sensor networks. It relies on a self-organizing *medium access control* (MAC) protocol, which uses an algorithm to decide the grade of participation of a sensor node to create a connected network based upon local information. On top of this connected network, a tightly integrated EYES Source Routing (ESR) is designed. It benefits from the local neighbor information of the MAC protocol and is able to efficiently maintain and recover routes based on the interaction with MAC protocol.

- *Reliable Cost-based Data-centric Routing (RCDR)*: The third contributions in the fourth quadrant of Figure 1.5 on page 12. It addresses the reliability of dynamic wireless sensor network in the point-multipoint routing scenarios by a data-centric approach. Current research in data-centric routing generally assumes a rather static network, leading to a strong performance degradation in a dynamic environment. A network wide reflooding of messages is the common solution to network topology changes. This contribution presents a Reliable Cost-based Data-centric Routing (RCDR) protocol for WSN. Instead of each sensor sending its own data report directly to the data sink, we introduce a global-local gradient paradigm to send only the aggregated data from the center of the event to the data sink. It ensures that sensors aggregate the collected data as close as possible to the data's origin, while only a small number of aggregated data are sent to the data sink. To increase the reliability of these aggregated data, they are sent via multiple adjustable routes to the data

sink. Local algorithms are designed to resume the network gradient when the network topology changes. The mobility of the data sink is especially solved with negative gradient. Furthermore, the movement of a monitored event, such as in the event tracking applications can also be efficiently handled.

1.3.2 Outline of the thesis

The chapters of this thesis are organized as follows.

Chapter 1: Introduction

Chapter 2: In this chapter, a state of the art survey in the routing of wireless ad hoc and sensor networks is presented.

Chapter 3: This chapter presents the first contribution of this thesis, a splitted multipath scheme to control the trade-off between traffic and reliability of data routing in wireless ad hoc and sensor networks.

Chapter 4: This chapter presents the second contribution of this thesis, a cross-layered approach for networking in wireless ad hoc and sensor networks.

Chapter 5: This chapter presents the third contribution of this thesis, a Reliable Cost-based Data-centric Routing (RCDR) protocol for wireless sensor networks.

Chapter 6: Conclusion and future works.

Chapter 2

State of the art on wireless routing protocols

This chapter introduces the state of the art routing protocols for wireless ad hoc and sensor networks, including a global view of the research in the area, a time line and taxonomy of the current routing protocols. This chapter also addresses the contributions of this thesis in relation to current research in routing for wireless networks.

2.1 Introduction

A wireless network is a computer network that uses wireless radio as its carrier or physical layer. It enables users to access information and services electronically, regardless of their geographic position. Wireless networks can be classified in two types: infrastructure-based networks and infrastructureless (ad hoc) networks. An infrastructure-based network consists of a network with fixed base stations which are connected by wires. The mobile hosts communicate with the base stations by wireless links, which enables them to move geographically within the communication radius of the base station. When the mobile host moves out of range of the connecting base stations, it connects with new base station for communication.

In contrast to infrastructure-based networks, ad hoc networks are self-configuring networks which consist of mobile hosts only connected by wireless links. In an ad hoc network, all nodes are free to move randomly and organize themselves arbitrarily. Minimal configuration and quick deployment make ad hoc networks very suitable for emergency situations like natural or human-induced disasters, meetings or conventions in which persons wish to quickly share information.

A wireless sensor network, a special kind of ad hoc network, consists of a number of sensors spread across a geographical area. Each sensor has wireless communication capability and sufficient intelligence for signal processing and networking of data. However, sensor nodes are constrained in energy supply and bandwidth. Such constraints, combined with a typical deployment of a large number of sensor nodes have posed many challenges to the design and management of sensor networks. These challenges necessitate energy-awareness at all layers of the networking protocol stack. The issues related to physical and link layers are generally common for all types of sensor applications. Therefore, the research on these areas has been focused on system-level power awareness such as dynamic voltage scaling [87] [28], radio communication hardware, low duty cycle issues, system partitioning and energy aware MAC protocols [106] [113] [89]. At the network layer, the main aim is to find methods for energy efficient route setup and reliable relaying of data from the sensor nodes to the sink so that the lifetime of the network is maximized.

Although wireless sensor networks share many similarities with wireless ad hoc networks, they have major constraints in energy supply and bandwidth. Due to this relationship between wireless ad hoc and sensor networks, this chapter introduces the current research effort on the routing of both networks and explains their differences. Figure 2.1 on page 16 gives a taxonomy of the various protocols introduced in this chapter. These protocols will be descirbed in Section 2.2 and Section 2.3.

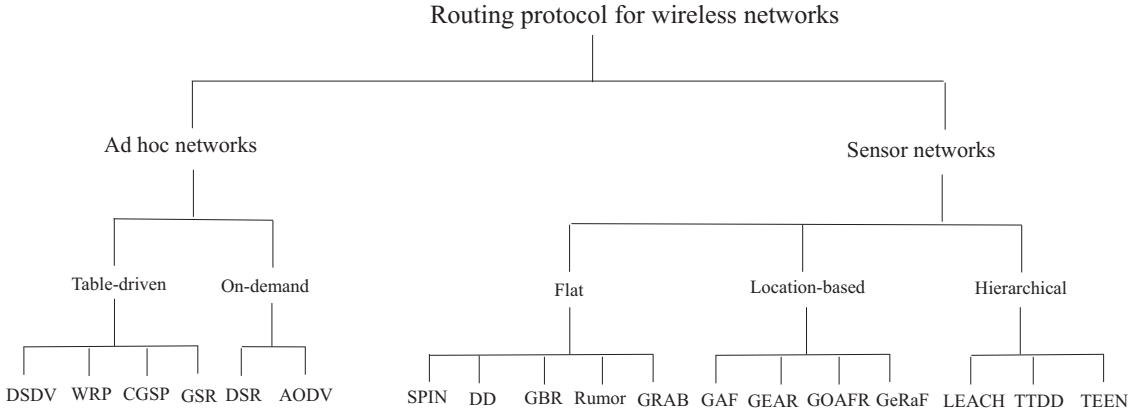


Figure 2.1: The taxonomy of the routing protocols for wireless networks

2.2 Routing protocols for wireless ad hoc networks

Wireless ad-hoc networks are autonomous systems of mobile nodes that form a network in the absence of any centralized support [75]. This is a new type of network and might be able to provide services at places where otherwise no communication is possible. The absence of a fixed infrastructure poses several types of challenges for the routing of this type of network. Routing protocols for ad hoc networks can be divided into two categories: table-driven (pro-active) and on-demand routing (reactive), based on when and how the routes are discovered.

2.2.1 Table-driven routing protocols

Table-driven routing protocols maintain a consistent and up-to-date view of the network. Each node uses routing tables to store the location information of other nodes in the network. This information is used to transfer data among various nodes. They respond to topology changes by propagating updates throughout the network in order to maintain a consistent network view. The differences between these protocols are the number of routing tables and the mechanism used to refresh the routing table.

The Destination-Sequenced Distance-Vector Routing (DSDV) [77] protocol is a table driven distance vector routing protocol algorithm that modifies the Bellman-Ford routing algorithm to include time stamps that prevent routing loop[1] formation. Global State Routing (GSR) [15] is similar to DSDV. It improves the idea of link state routing[2] by avoiding flooding of routing messages. The Wireless Routing Protocol (WRP) [67] is a another distance vector routing protocol, which belongs to the class of path-finding algorithms that exchange second-to-last hop to destinations in addition to distances to destinations. This extra information helps remove the "counting-to-infinity" problem[3] that vex most distance vector routing algorithms. It also speeds up route convergence when a link failure occurs. Cluster-head Gateway Switch Routing (CGSR) [16] is also based on DSDV. It uses a Least Cluster Change (LCC) clustering algorithm to reduce the routing table size by keeping only one entry for one whole destination cluster.

[1] A network problem in which packets continue to be routed in an endless circle.

[2] The basic concept of link-state routing is that every node receives a map of the connectivity of the network, in the form of a graph showing which nodes are connected to which other nodes. Each node then independently calculates the best next hop from it for every possible destination in the network. The collection of best next hops forms the routing table for the node.

[3] It happens when adapting to changes in topology, Destination networks which become unreachable will be advertised with increased distance until their distance reaches infinity.

However, table-driven protocols might not be considered an effective routing solution for a dynamic ad-hoc network. It is very slow to converge and may be prone to routing loops. Nodes in mobile ad-hoc networks operate with low battery power and limited bandwidth. Presence of high mobility, large routing tables and low scalability result in consumption of both bandwidth and battery life of the nodes. Moreover, table-driven protocols keep a constant overview of the network. This could create unnecessary overhead as they may react to changes in the network topology even if no traffic is affected by the topology modification.

2.2.2 On-demand routing protocols

On-demand routing protocols were designed with the aim of reducing control overhead, thus increasing bandwidth and conserving power at the mobile stations. In contrast to table-driven routing protocols, the routes are only created as and when required. There is no updating of every route table in the network. Instead, it only maintains the freshness of routes that are being used or being set up. If a source node requires a route to the destination for which it does not have route information, it starts a route discovery process, which goes from one node to the other until it arrives at the destination or a node in-between has a route to the destination. The established route is maintained by a route maintenance procedure until either the destination becomes unreachable or the route is no longer desired. These protocols limit the amount of bandwidth consumed by maintaining routes to only those destinations for which a source has data traffic. The following texts introduces two of the better known on-demand protocols.

Dynamic source routing protocol(DSR)

The Dynamic Source Routing protocol (DSR) [44] [43] is a source-routed on-demand routing protocol designed specifically for wireless ad hoc networks. The nodes can dynamically discover a source route across multiple network hops to any destination in the network. This makes the network completely self-organizing and self-configuring without the need for a network infrastructure or administration

The DSR protocol is composed of the two mechanisms “Route Discovery” and “Route Maintenance”, which work together to allow the discovery and maintenance of source routes in the ad hoc network. When a source node attempts to send a packet to a destination node and does not already know a route to the destination node, it initiates Route Discovery by broadcasting a route request. The route request message identifies the source and destination of the Route Discovery, which also contains a unique request ID, assigned by the source node. The header of the route

request also contains a record listing the address of each intermediate node through which this particular copy of the route request message has been forwarded. A route reply is generated when either the destination node or an intermediate node with a route to the destination receives the route request packet. Then, the source can send data packets with the complete, ordered list of nodes through which it must pass, avoiding the need for up-to-date routing information in the intermediate nodes. Route Maintenance enables the source node to detect that the network topology has changed and it can no longer use its route to the destination, because a link along the route no longer works. When that happens, the source node can attempt to use any other route it knows to the destination, or can invoke Route Discovery again to find a new route.

DSR has a very low routing overhead because it operates entirely on demand. Morevoer, the whole routing information is contained in the packet header and does not periodically update the topology within the intermediate nodes in the network. For example, DSR does not use any periodic routing advertisement, link status sensing or neighbor detection packets. DSR performs well for small to medium sized networks as its overhead can be very small. However, the overhead does increase for networks with larger hop diameters as more intermediate addresses need to be contained in the packet headers.

Ad hoc on-demand distance vector routing(AODV)

Ad hoc On-demand Distance Vector Routing (AODV) [78] [76] is an on-demand version of the table-driven Dynamic Destination-Sequenced Distance-Vector (DSDV) protocol [77]. It builds routes between nodes only when requested by source nodes, as opposed to DSDV, which maintains the list of all routes.

AODV sets up routes by a route request and route reply cycle. When a source node requires a route to a destination, it broadcasts a route request packet across the network. Intermediate nodes update their information for the source node and set up backwards pointers to the source node in the route tables. This information is used to form the reply path for the route reply packet. This broadcast message propagates in the network until it reaches the destination or an intermediate node that already has a route to the destination. This intermediate node or the destination then initiates the reply cycle by sending a route reply back to the source node. As the route reply travels back to the source, intermediate nodes set up forward pointers to the destination. Once the source node receives the route reply, it begins to forward data packets to the destination. As long as the route remains active, it will continue to be maintained by AODV. Once the source stops sending data packets, the links will time out and eventually will be deleted from the intermediate node routing tables. If a link fails while the route is active, the nodes upstream of the break send

a route error message to the source node. After receiving the error message, it can reinitiate route discovery to the destination if needed.

Furthermore, AODV can setup multicast routes in a similar manner. It forms trees which connect multicast group members and the nodes needed to connect the members. AODV uses sequence numbers to ensure the freshness of routes. The advantage of AODV is that it is simple, loop-free, self-starting and scales for large numbers of mobile nodes. Moreover, it does not create extra traffic for communication along existing links.

2.3 Routing protocols for wireless sensor networks

Wireless sensor networks have theirs own unique characteristics which create new challenges for the design of routing protocols for these networks. First, sensors are very limited in transmission power, computational capacities and most of all, in energy. Thus, the operating and networking protocol must be kept much simpler as compared to other ad hoc networks. This simplicity may also break with conventional layering rules for networking protocols, since abstractions typically cost time and space. Second, due to the large number of application scenarios for WSNs, it is unlikely that there will be a one-thing-fits-all solution for these potentially very different possibilities. The design of a sensor network routing protocol changes with application requirements. Third, data traffic in WSNs has significant redundancy since data is probably collected by many sensors based on a common phenomenon. Such redundancy needs to be exploited by the routing protocols to improve energy and bandwidth utilization. Fourth, in many of the initial application scenarios, most nodes in WSNs were generally stationary after deployment. However, in recent development, sensor nodes are increasingly allowed to move and change their location to monitor mobile events, which results in unpredictable and frequent topological changes.

Due to such different characteristics, many new protocols have been proposed to solve the routing problems in WSNs, as given in [5] [3]. These routing mechanisms have taken into consideration the inherent features of WSNs, along with the application and architecture requirements. To minimize energy consumption, routing techniques proposed in the literature for WSNs employ some well-known ad hoc routing tactics, as well as, tactics special to WSNs, such as data aggregation and in-network processing, clustering, different node role assignment and data-centric methods. In the following text, we introduce current research on routing protocols for wireless sensor networks according to the network structure as flat, location-based and hierarchical, based on the taxonomy in [5] [3].

2.3.1 Flat routing protocols

In flat routing protocols, the sensor nodes are all peers of each others. Each node typically plays the same role and they collaborate together to perform the sensing task. In the rest of this subsection, we summarize these flat routing protocols and highlight their advantages and performance issues.

Sensor Protocols for Information via Negotiation (SPIN)

In [52], the author proposed a family of adaptive protocols called Sensor Protocols for Information via Negotiation (SPIN) which efficiently disseminate information in a wireless sensor network. The SPIN family of protocols uses data negotiation and resource-adaptive algorithms. Nodes running SPIN assign a high-level name to their data, called meta-data, and perform meta-data negotiations before any data is transmitted. These negotiations ensure that redundant data is not sent throughout the network. Moreover, SPIN has access to the current energy level of the node and adapts the protocol it is running based on how much energy is remaining.

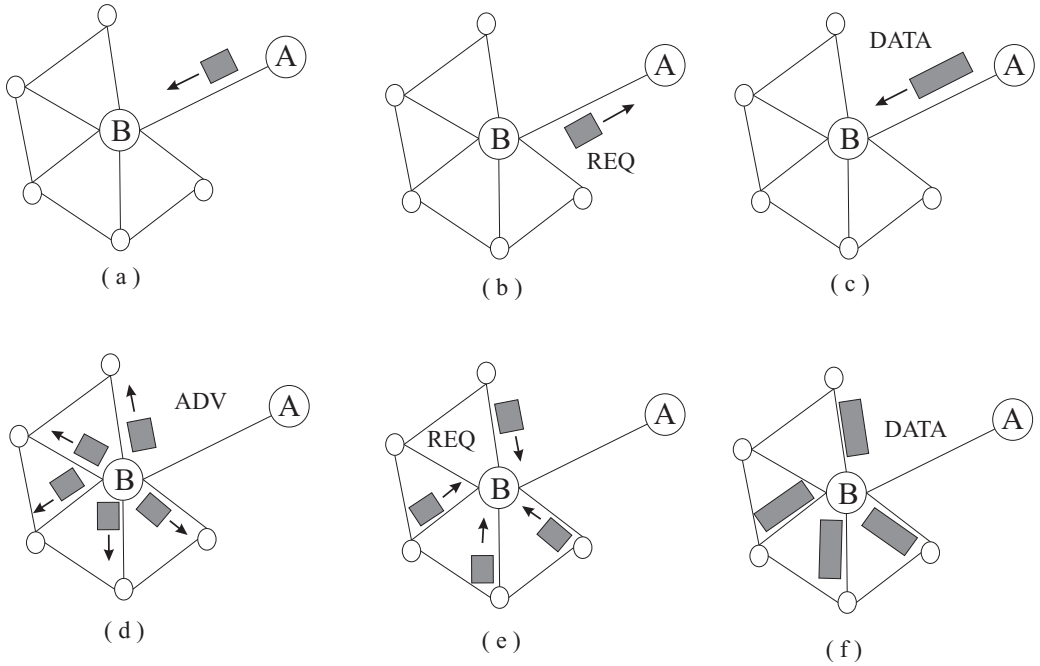


Figure 2.2: SPIN Protocol. Node A starts by advertising its data to node B (a). Node B responds by sending a request to node A (b). After receiving the requested data (c), node B then sends out advertisements to its neighbors(d), who in turn send requests back to B (e-f).

As conventional data dissemination approaches, flooding[4] and gossiping[5] waste valuable communication and energy resources by sending redundant information throughout the network. SPIN's meta-data negotiation solves this classic problem and achieves more energy efficiency by reducing the number of redundant transmissions. SPIN is a three-stage protocol, as sensor nodes use three types of messages ADV, REQ and DATA to communicate as shown in Figure 2.2 on page 21, which is redrawn from source [52]. ADV is used to advertise new data in the form of meta-data, REQ to request the specific data, and DATA is the actual message itself. When a SPIN node wants to share data with other nodes, it first broadcasts an ADV message describing the DATA it wants to send. If a neighbor is interested in the data, it sends a REQ message for the DATA and the DATA is sent to this neighbor node. The neighbor sensor node then repeats this process with all its neighbors. As a result, the entire sensor area will receive a copy of the data.

Simulation results in [52] show that SPIN is more energy efficient than flooding or gossiping, while distributing data at the same rate or faster than either of these protocols. One of the advantages of SPIN is that topological changes are localized since each node only needs to negotiate with its single-hop neighbors. However, this does not guarantee the delivery of data to a node that is more than two hops away from the source node and is interested in the data. This happens when all the intermediate nodes between the sources and this node are not interested in the data.

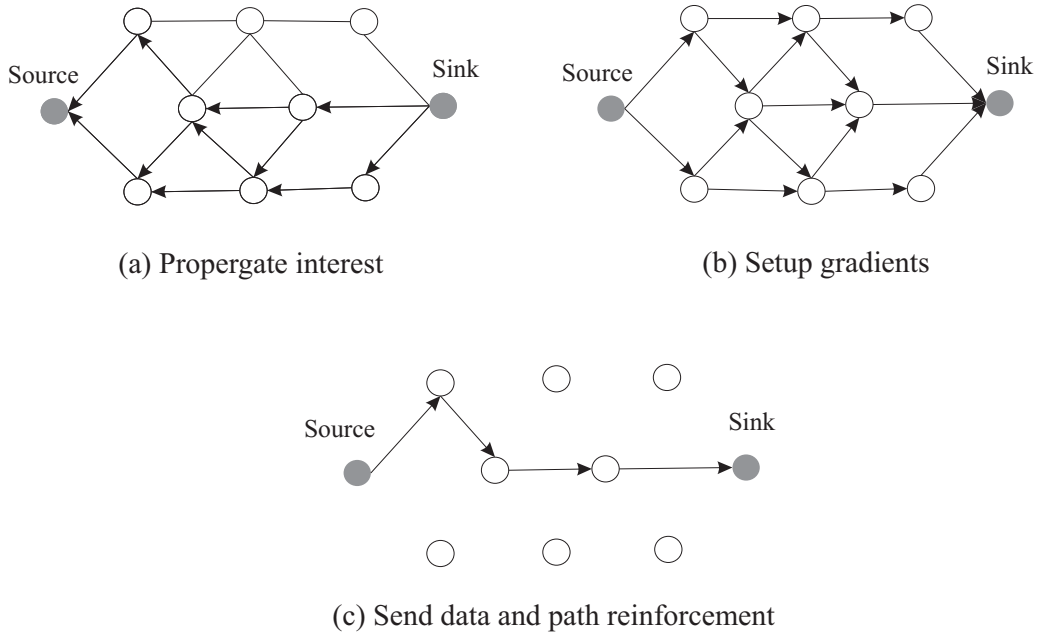
Directed Diffusion (DD)

Directed Diffusion [41] (DD) is a popular data aggregation paradigm for wireless sensor networks. It is a data-centric and application-aware paradigm, in the sense that all data generated by sensor nodes is named by attribute-value pairs. Such a scheme combines the data coming from different sources enroute to the sink by eliminating redundancy and minimizing the number of transmissions. In this way, it saves the energy consumption and increases the network lifetime of WSNs.

In Directed Diffusion, the base station requests data by broadcasting interests, which describes a required task to be implemented by the network. The interest is defined using a list of attribute-value pairs such as name of objects, interval, duration and geographical area. Each node receiving the interest can cache it for later use. As the interest is broadcasted through the network hop-by-hop, gradients are setup to draw data satisfying the query towards the requesting node. A gradient is a reply link to the neighbor from which the interest was received. It contains the

[4]Each node in the network rebroadcasts every received packet not destined to itself.

[5]Each node forwards a message with some probability, to reduce the overhead of the routing protocols.



(a) Propergate interest

(b) Setup gradients

(c) Send data and path reinforcement

Figure 2.3: Three phases of Directed diffusion protocol

information derived from the received interest's fields, such as the data rate, duration and expiration time. Each sensor that receives the interest, sets up a gradient toward the sensor nodes from which it received the interest. This process continues until gradients are setup from the sources all the way back to the base station. In this way, several paths can be established, so that one of them is selected by reinforcement. The sink resends the original interest message through the selected path with a smaller interval, hence reinforcing the source node on that path to send data more frequently. Figure 2.3 on page 23 redrawn from [5] shows examples of Directed Diffusion ((a) sending interests, (b) building gradients and (c) data dissemination).

Directed Diffusion has many advantages. Since Directed Diffusion is data centric, all communication is aimed at neighbor-to-neighbor communication without the need for a node addressing mechanism. Each node can cache and process the data in addition to sensing. Caching is a big advantage in terms of energy efficiency, delay, robustness and scalability of coordination between sensor nodes, which is the essence of the data diffusion paradigm. The nodes are also able to perform data aggregation to further reduce the redundancy, which is modelled as a minimum Steiner tree problem [49]. However, Directed Diffusion may not be applied to sensor network applications that require continuous data delivery to the base station, such as environmental monitoring. This is because the query driven on demand data model does not help in this regard. In addition, the naming schemes used in Directed Diffusion are application dependent and should be each time defined. Moreover,

extra energy is required for constantly updating messages and matching data to queries .

Gradient-based Routing (GBR)

Gradient-Based Routing [84] is a another version of Directed Diffusion, which aims to distribute traffic evenly throughout the network in order to increase the network lifetime. The key idea is to memorize the number of hops when the interest is diffused through the whole network. As such, each node can calculate a parameter called the height of the node, which is the minimum number of hops required to reach the base station. The difference between a node's height and that of its neighbor is considered the gradient on that link. A packet is forwarded on a link with the largest gradient.

GBR uses some additional techniques such as data aggregation and traffic spreading in order to uniformly divide the traffic over the network. When multiple paths pass through a node, that node may combine data according to a certain function. In GBR, three different data dissemination techniques have been discussed:

- *Stochastic scheme*: a node picks one gradient at random when there are two or more next hops that have the same gradient.
- *Energy-based scheme*: a node increases its height when its energy drops below a certain threshold, so that data are diverted to the node with a high amount of energy remaining.
- *Stream-based scheme*: new streams are not routed to nodes that are currently on the path of other streams

The main objective of these schemes is to obtain a balanced distribution of traffic in the network, avoiding parts of the network that are energy depleted and thus increasing the overall network lifetime. Simulation results of GBR in [84] showed that GBR outperforms Directed Diffusion in terms of total energy used for communication.

Rumor routing protocol

Rumor routing [9] is another wireless sensor network routing algorithm, which aims at lower energy consumption. Instead of flooding the whole network with queries, it spreads the information of an event to other nodes in the network, thus enabling queries to discover paths to the events.

Rumor routing tries to maintain the trade-off between query flooding and event flooding. The main idea is to create paths leading to each event as the event happens, and later route the queries along these paths. It uses agents to create paths leading to each event when the event happens. The agents are actually long-lived messages traversing in the network. Later, queries can be routed along these agent-generated paths. In order to join the path, the queries are sent on a random walk in the network until it finds the event path, instead of flooding it throughout the network. Each node in the network maintains a list of its neighbors and an event table with forwarding information to all known events. As the agent travels in the network, it can combine its own event table with event tables of visited nodes along its path.

Rumor routing is a tunable and more energy efficient algorithm than flooding-based ones, especially when geographic information is not available. Its usefulness depends on how well the configuration parameters are set for the particular event and query distribution in the network. Simulation results shows that the algorithm handles node failures and allows for tradeoffs between setup overhead and delivery reliability. However, the algorithm only performs well when the number of events is small. The cost of maintaining agents and event-tables in each node increases with the number of events.

GRAdient Broadcast(GRAB)

GRAdient Broadcast (GRAB) [111] is another protocol built on Directed Diffusion [41]. The basic idea is to deliver the data packets issued by a source sensor along the direction of a sink by decreasing some costs, which are initially built and maintained by the sink but kept by each sensor.

GRAB first builds and maintains a cost field, providing each sensor the direction to forward sensing data. The cost at a node is the minimum energy overhead to forward a packet from itself to the sink along a path. The cost field implicitly states the global direction towards the sink. Sensors closer to the sink will have smaller energy overhead costs. When a node forwards a packet, it simply includes its own energy cost in the packet. On receiving this packet, neighbors will participate in the packet forwarding process only when its own cost is smaller than that of the previous sender. Because multiple paths of decreasing cost may exist, the data is then forwarded by a band of an interleaved mesh from each source to the receiver. GRAB allows the sender to adjust the robustness of data delivery by controlling the width of the forwarding paths by the amount of credit carried in each data message. This design harnesses the advantage of large scaling and increases the reliability of data delivery by collective efforts of multiple nodes, without dependency on any one individual node.

Simulation results show that interleaved mesh performs better than multiple parallel paths. However, GRAB requires each node's cost value to be periodically refreshed by a sink, causing the same problem of excessive overhead as flooding. Moreover, the mobility of a sink causes a network reflooding to reinitiate setup of the cost. As a result, GRAB may not work well in dynamic network topology or even in semi-static ones.

2.3.2 Location-based routing protocols

More recently, there has been a growing focus on a class of routing algorithms that make routing decisions based on cached geographical information of neighboring sensor nodes to find a relatively optimal path without flooding. These algorithms improve network scalability by reducing the total routing overhead. The idea is to use location information to reduce propagation of control messages, control packet flooding and make simplified packet forwarding decisions. Undoubtedly, the Global Positioning System (GPS) is the most well-known location service in use today. However, the GPS approach is relatively expensive for low-cost, ad-hoc sensor networks, since GPS is based on extensive infrastructure and not available in special environments such as indoors. Alternatively, information such as connectivity, incoming signal strength, time of flight, angle of arrival, etc. are successfully used to determining the position of sensor node with only localized computations [88] [83] [19]. The following sections will discuss several current location-based routing protocols for wireless sensor networks.

Geographic Adaptive Fidelity(GAF)

Geographical Adaptive Fidelity (GAF) [110] self-configures redundant nodes into small groups based on their locations and uses localized, distributed algorithms to control node duty cycle to extend network operational lifetime. GAF addresses this problem by dividing the whole area where nodes are distributed into small virtual grids as shown in Figure 2.4 on page 27 redrawn from [110]. In each grid all nodes are equivalent for routing and coordinate with each other to determine the sleep schedule. Nodes are then able to achieve load balancing by periodically waking up.

GAF determines node equivalence based on location information and virtual grids. Each node uses its location information to associate itself within a point in the virtual grid. Nodes associated with the same point on the grid are equivalent from a routing perspective. Such equivalence is used to keep some of the redundant nodes located in the same grid area in sleeping state in order to save energy and increase the network lifetime as the network size increases. GAF also uses a load balancing

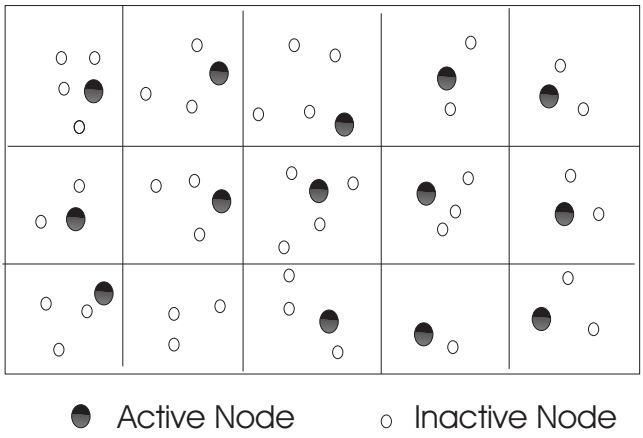


Figure 2.4: Virtual grid and active nodes in GAF

strategy to increase the network lifetime. It tries to rotate the role of active node in each grid periodically. In high mobility cases, the leaving node informs its neighbor in the grid about its estimated leaving time, so that neighboring nodes can adjust their sleeping schedule to accommodate the changes.

Simulation results show that GAF performs at least as well as a normal ad hoc routing protocol in terms of latency and packet loss. It also extends network lifetime proportionally to node density. However, GAF assumes that sensor nodes know their locations through a GPS receiver, which is inconceivable in the current technology on a sensor node.

Geographic and Energy Aware Routing (GEAR)

Geographical Energy Aware Routing (GEAR) [115] is an energy aware geographic protocol to increase the lifetime of sensor networks. It uses geographic information to limit the query flooding in Direct Diffusion [41] by only sending interest to a certain region rather than the whole network. GEAR uses energy aware metrics for neighbor selection in such a way that each node tries to balance the energy consumption among its neighbors. Each node keeps an estimated cost, which is a combination of residual energy and distance to the destination through its neighbors. The node also maintains a learned cost, which is the propagated cost that accounts for the real network topology, especially the routing holes. A node encounters a routing hole when no other neighbors are closer to the destination than itself. In this situation, the learned cost is larger than the estimated cost. The estimated cost is adjusted when the learned cost is sent one hop back every time a packet reaches the destination.

When forwarding the data from the source to the destination, two phases exist in GEAR. First, when forwarding packets towards the target region, the forwarding sensor node chooses one of the neighbors closest to the target region as its next hop. If a routing hole occurs, one of the neighbors is picked to forward the packet based on the learning cost function. Second, when the data reaches the target region, the packet can be diffused by two means, recursive geographic forwarding or restricted flooding. When the target region node density is low, restricted flooding can be used to forward the data to the destination. Or, when the target region node density is high, recursive geographic flooding can be used to save energy by dividing the target region into sub regions and sending one copy to each sub-region. This splitting and forwarding process continues until the destination is reached.

The simulation results show that GEAR not only reduces energy consumption for the route setup, but also performs better in terms of packet delivery. However, GEAR is sensitive to location errors and it increases the average path length for a packet. Although GEAR decreases the number of states a node should keep, it requires a location service to map locations and node identifiers, which is considered too expensive for wireless sensor networks.

Greedy Other Adaptive Face Routing (GOAFR)

Greedy Other Adaptive Face Routing (GOAFR) [51] is a geometric ad-hoc routing algorithm combining greedy and (other adaptive) face routing. GOAFR is proven to be both asymptotically optimal and efficient on average-case graphs.

In the greedy mode, GOAFR uses the closest-to-destination method to choose the next node. However, it can get easily stuck at a routing hole, i.e. no neighbor is closer to a node than itself. When the greedy mode gets stuck, a face-routing algorithm is used as a backup mode to bypass the routing hole. Face routing is an algorithm that routes exclusively along paths of a planar graph. It first computes a planar subgraph of the underlying wireless connectivity graph, then defines a consistent forwarding mechanism for routing around "routing holes".

The backup mode is an Other Face Routing (OFR) algorithm which is a variant of Face Routing (FR) [92]. The face routing works on the planar graph and routes along the face boundaries. It guarantees successful routing if the source and the destination are connected. However, the worst-case cost of FR is proportional to the number of nodes in the network. An Adaptive Face Routing (AFR) algorithm tries to compete with the best route in the worst-case scenarios. The key idea is to use an ellipse to restrict the searchable area during routing so that in the worst case, the total cost is no worse than a constant factor of the cost for the optimal route. Although AFR is asymptotically worst-case optimal under a lower bound argument,

it is not average-case efficient. On the other hand, OFR can find the best node on a series of face boundaries by using geometric planes, which makes it efficient on average-case graphs.

GOAFR is the first worst-case optimal and average-case efficient algorithm, which percolates theory to evaluate routing algorithms. Simulation on different cost metrics shows that the algorithm behaves well in dense networks, as opposed to AFR. But it fails for very simple configurations as shown in [51]. Moreover, the algorithm should find ways to further improve the average case performance.

Geographic Random Forwarding (GeRaF)

Geographic Random Forwarding (GeRaF) [118] is a recent transmission scheme based on geographical routing where packets are relayed on a best-effort basis, i.e., the actual node which acts as a relay is not known in advance by the sender, but rather is decided after the transmission has taken place.

This idea relies on the fact that in the wireless environment broadcast, is free (from the sender's point of view) and that in the presence of randomly changing topologies, a node may not be aware of which of its current neighbors is in the best position to act as a relay. Since the intended recipient is not specified, multiple nodes may be able to receive the packet and a *receiver contention scheme* is therefore needed to guarantee that a single relay is chosen, thereby avoiding packet duplication.

GeRaF assumes that each node has some knowledge of its own position and of the sink node position . Once a node has a packet to send, it sends it using some type of broadcast address while specifying its own location and the location of the intended destination. All active nodes in the coverage area will receive this packet and assess their own priority in trying to act as a relay, based on how close they are to the destination.

Simulation results show that GeRaF does not address a specific node, which allows it to decrease the duty cycle of each node without increasing latency if the node density is adequate. It also has significantly fewer hops than GAF [110] and therefore a smaller energy consumption for the delivery of a packet. However, GeRaF needs knowledge about the sink location, which may become a problem if the sink moves. Also, the protocol is designed to work well in the presence of a dense node deployment, but may fail if the topology is sparse.

2.3.3 Hierarchial routing protocols

Hierarchical routing protocols group routers together by function into a hierarchy. A hierarchical protocol allows to make best use of the fast powerful routers as backbone routers, and the slower, lower powered routers may be used for access purposes.

Low Energy Adaptive Clustering Hierarchy (LEACH)

Low Energy Adaptive Clustering Hierarchy (LEACH) [35] is one of the first hierarchical routing approaches for sensors networks, which is a cluster-based protocol. LEACH organizes sensor nodes into clusters based on received signal strength and uses local cluster-heads as both routers to the base station and as a data fusion point.

LEACH is a cluster based approach where the cluster head is elected with a probability based on the amount of energy left in the node. A non-cluster head node joins the cluster which requires the minimum communication energy to the cluster head. Communication with the cluster head is done via TDMA MAC, which has a fixed schedule for communicating with non-cluster nodes. The non-cluster heads can go into sleep mode if according to the schedule, it is not in transmission. A cluster head is the data aggregation point and it communicates directly with the sink or user. The cluster head compresses correlated data from its cluster members and sends aggregated packets to the sink in order to reduce the amount of information that must be transmitted via radio. LEACH also randomly rotates the cluster head to evenly spread the load among cluster members.

LEACH assumes that the sources and users are stationary and events monitored are continuous. The interest propagation in the network is predetermined and the data dissemination mechanism is broadcasting. LEACH optimizes energy used by shutting down node's radios and balancing the load among cluster members. In hierarchial routing, only two hops are needed for reaching the sink. The distributed hierarchical approach makes it scalable. However, the failure of a cluster head introduces a problem to the algorithm. Furthermore, cluster head selection is difficult to optimize, which brings extra overhead. It also makes an expensive assumption that all nodes are capable of long range communication

The idea proposed in LEACH has been an inspiration for many hierarchical routing protocols. LEACH with negotiation was proposed in [32]. The main extension is a meta-data negotiation between the cluster head and its members before data transfer, which ensures that only new data is transmitted to the cluster head. Power-Efficient GAthering in Sensor Information Systems (PEGASIS) [54] is another protocol based on the LEACH protocol. The key idea is to form a chain

among the sensor nodes instead of clusters, so that each node will communicate with a close neighbor. Gathered data moves from node to node, gets fused and eventually a designated node transmits to the sink. Nodes rotate transmitting to the sink so that the network lifetime is increased. Hierarchical-PEGASIS [55] is an extension to PEGASIS, which decreases the delay during transmission from the designated node to the sink. Although the PEGASIS approach avoids the clustering overhead of LEACH, they still require dynamic topology adjustment which also introduces overhead.

Threshold-sensitive Energy Efficient sensor Network protocol (TEEN)

Threshold-sensitive Energy Efficient sensor Network protocol (TEEN) [58] is a hierarchical protocol proposed for time-critical applications. TEEN pursues a hierarchical approach along with the use of a data-centric mechanism. The sensor network architecture is based on a hierarchical grouping where closer nodes form clusters and this process continues on the second level until the sink is reached, as shown in Figure 2.5 on page 31 redrawn from [58] .

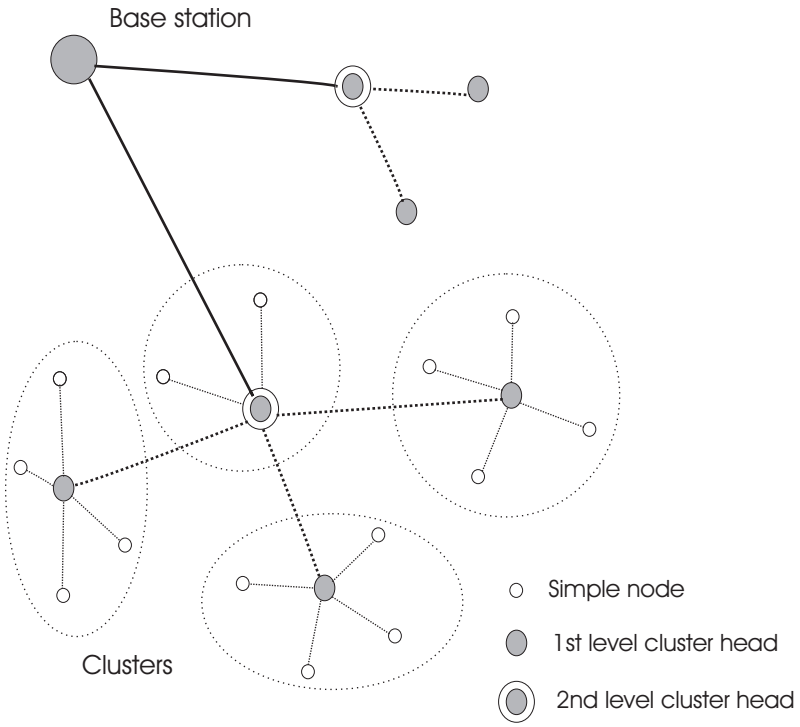


Figure 2.5: Two level clustering in TEEN

After clusters are formed, the cluster head broadcasts two thresholds to the nodes. Hard threshold is the minimum possible value of an attribute to trigger a

sensor node to transmit to the cluster head. It reduces the number of transmissions significantly by only transmitting when the sensed attribute is in the range of interest. Once the data values are transmitted, a node only transmits when the value of that attribute changes over the soft threshold. The soft threshold further reduces the number of transmissions that might have otherwise occurred when there is little or no change in the sensed attribute. The two thresholds can be tuned by the user, which controls the trade-off between energy efficiency and data accuracy. The main drawback of this scheme is that the protocol is not good for applications where periodic reports are needed. Moreover, if the thresholds are not received or not reached, the nodes will never communicate and the sink will not get any data from the network at all.

The Adaptive Threshold sensitive Energy Efficient sensor Network protocol (APTEEN) [59] is an adaptive extension to TEEN, which captures both periodic data collections and reacts to time-critical events. When the clusters head forms the clusters it sends a transmission schedule to all nodes, in addition to the attributes and the threshold values. If a node does not send data for a time period equal to the count time, it is forced to sense and retransmit the data.

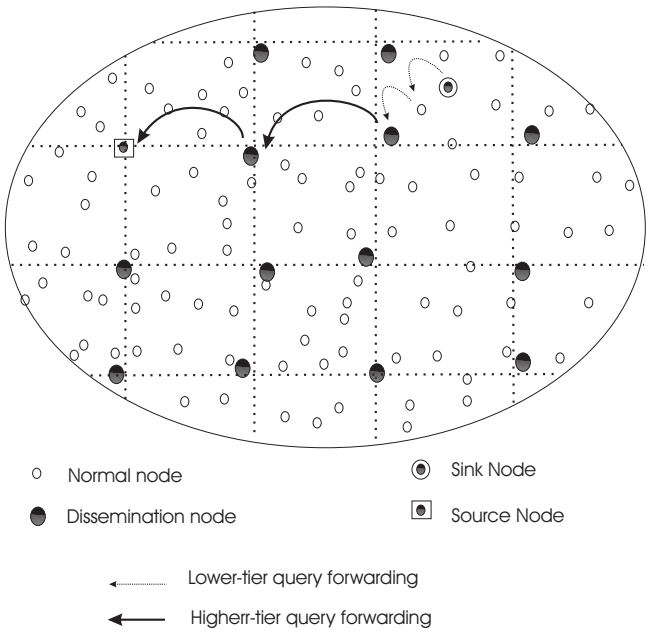


Figure 2.6: Two-tier grid structure in TTDD

Simulation results shows that TEEN and APTEEN outperform LEACH [35]. The experiments have demonstrated that APTEENs performance is somewhere between LEACH and TEEN in terms of energy consumption. The main disadvantage of the two protocols are the overhead with forming and maintaining clusters at

two levels, as well as the complexity associated with implementing threshold-based functions, and how to deal with attribute-based naming of queries.

Two-Tier Data Dissemination (TTDD)

Two-Tier Data Dissemination (TTDD) [112] provides data delivery to multiple mobile base-stations based on a decentralized architecture. It assumes there are homogeneous sensors, each aware of its own location and generally stationary. There are multiple sources and mobile sinks, which query the network to collect sensing data. Instead of broadcasting their location information to all sensor nodes, TTDD uses a two-tier data dissemination model to deal with the sink mobility problem and reduce energy consumption.

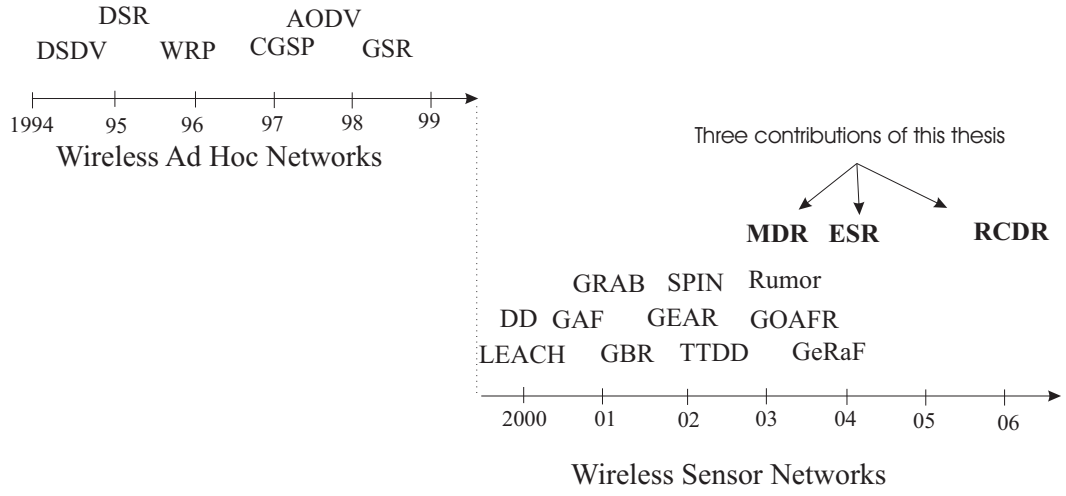


Figure 2.7: The development timeline for the routing protocols of wireless networks

In TTDD, each data source uses a grid structure to divide the topology into cells as shown in Figure 2.6 on page 32 redrawn from [112]. Only sensors located at a cell boundary need to forward the data. The data sink proactively builds the two-tier grid structure throughout the network and sets up forwarding points in the sensors closest to the grid boundary, which are called dissemination nodes. The lower tier is the cell at the sink's current location and the higher tier contains the dissemination nodes at cell boundaries. The data sink only floods the query within its own cell. When the nearest dissemination node in the cell receives the query, it forwards it to its adjacent dissemination node in another cell. This process continues until the query reaches the producer or one of the dissemination nodes that have the corresponding data. During the query propagation, the network establishes the reverse path towards the sink, so that the data could be forwarded on the same path as that of the query propagation.

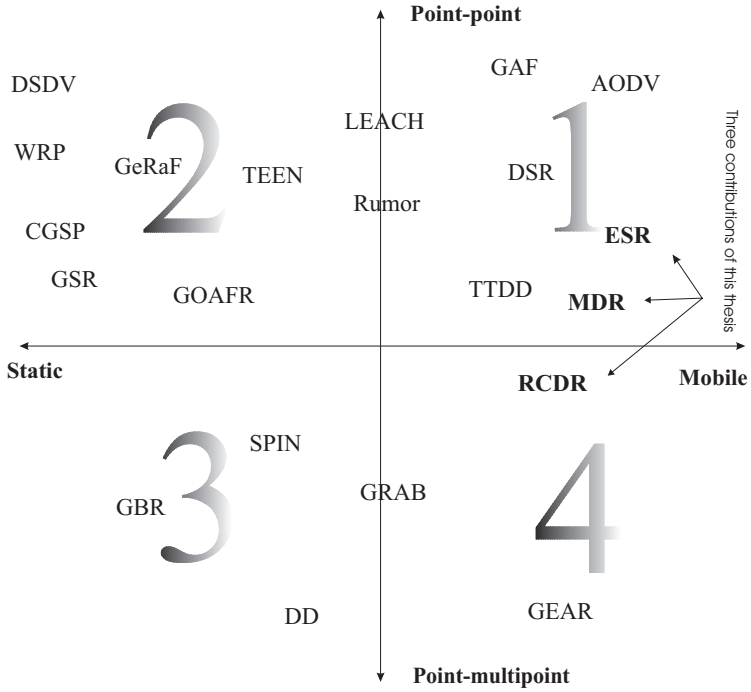


Figure 2.8: The communication patterns and various wireless routing protocols

TTDD localizes the impact of sink mobility on data forwarding. Only a small set of sensor nodes (dissemination nodes) need to maintain the forwarding state. Simulation results between TTDD and Directed Diffusion showed that TTDD can achieve longer lifetimes and data delivery delays. However, the overhead associated with maintaining and recalculating the grid as the network topology changes may be high. Furthermore, TTDD assumes the availability of a very accurate positioning system which is not yet available for WSNs.

Simulation results show that TTDD can build an infrastructure in stationary sensor networks to reduce the influence of sink mobility based on location information. It achieves a longer network lifetime than Direct Diffusion. However, the overhead to maintain the two-tier grid structure is high when mobility increases and an accurate location information for each sensor is still too expensive for WSNs.

2.4 Conclusion

In this chapter, we introduced various research on the routing of wireless ad hoc and sensor networks. Figure 2.7 on page 33 shows the development time line for these routing protocols and the three contributions of this thesis. The research for

wireless ad hoc network began in 1994. And since 2000, more research effort has been focused on the development of routing algorithm for wireless sensor networks.

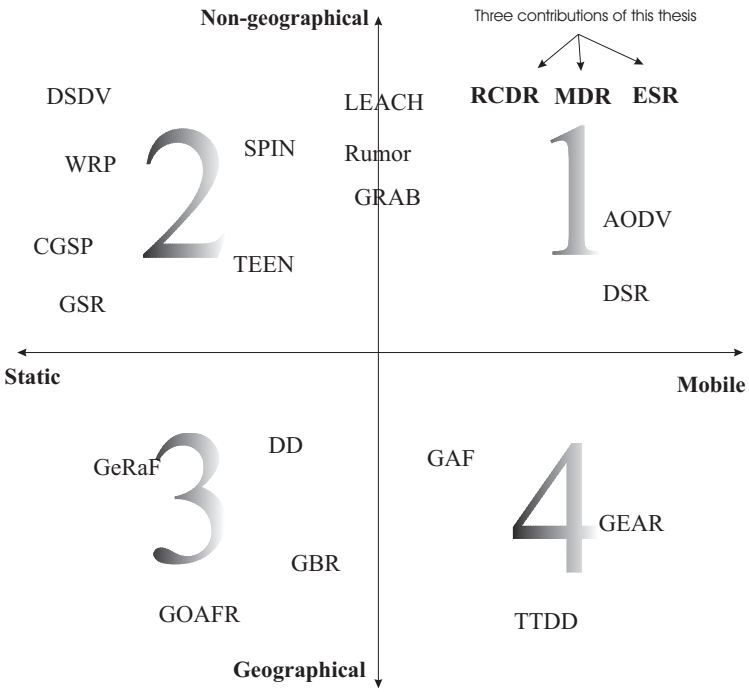


Figure 2.9: The geographical characteristic of various wireless routing protocols

In Figure 2.8 on page 34, we classify the introduced wireless protocols in this chapter according to their proposed application dynamics and communication patterns. The graph illustrates that the contributions of this thesis are in the dynamic aspect of the wireless sensor network. Although some research has already been done in this area, the uniqueness of this thesis is found in Figure 2.9 on page 35, where the wireless protocols are classified into another dimension. It shows that most of the current research for dynamic networks has been based on geographical information. However, considering the tiny, cheap and resource limited sensor node, additional hardware, such as GPS, is relatively expensive and energy intensive. In this thesis, we focus on the reliability and energy efficiency of routing protocol for dynamic wireless sensor networks without geographical information support.

Chapter 3

Reliable splitted multipath routing protocol

In a Wireless Sensor Network (WSN), sensor nodes are deployed unreliable, particularly in harsh and unpredictable environments. Sensor nodes are prone to failures. They may have many failure modes, such as radio link failure, battery depletion, environmental interference and physical damage, each one of which decreasing the performance of the network. In WSN, the reliability of the routing protocols can be increased by providing several paths from the source node to the destination node and by sending the same packet through each of them (the algorithm is known as multipath routing*). Using this technique, the traffic increases significantly. In this chapter, we analyze the combination of a new multipath routing mechanism and a data-splitting scheme that results in an efficient solution for achieving high delivery ratios while keeping the traffic at a low value. It addresses the reliability of a dynamic wireless sensor network in the point-point routing scenarios by multipath routing. The algorithms presented will assure that the gathered data reaches its destination in the network by assuming that nodes may not be available during the routing procedure. Additional energy will be required only for a small amount of computations; this is almost negligible compared with the energy used for communications [81]. Simulation results are presented in order to characterize the performance of the algorithm. Our major contribution in this chapter is a new multipath routing algorithm with a data splitting scheme. The work done in this chapter is published in two separate papers of WCNC'03 [22][1] and NPC'04 [107][2].*

[1]Stefan Dulman, Tim Nieberg, Jian Wu, and Paul Havinga. "*Trade-off between traffic overhead and reliability in multipath routing for wireless sensor networks*". In Proceedings of the Wireless Communications and Networking Conference, 2003.

[2]Jian Wu, Stefan Dulman and Paul Havinga. "*Reliable splitted multipath routing for wireless sensor network*". In Proceedings of IFIP International Conference on Network and Parallel

3.1 Introduction

In wireless sensor networks, the application sometimes requires to disseminate a large amount of bulk data to the destination with high reliability and low delay. In examples like streaming or code dissemination, the application needs to release large amounts of data which are composed of large packets to the remote party. These transmissions are performed over the wireless channel, which is known as a low-bandwidth and lossy medium. Therefore, reliability issues need to be addressed. Moreover, as the network diameter grows, data generated by one or more sources usually has to be routed through several intermediate nodes to reach the destination due to the limited range of the node's wireless transmission. Problems arise when intermediate nodes fail to forward the incoming messages either due to mobility or node failures. Usually, acknowledgements and retransmissions are implemented to recover the lost data. However, this generates a large amount of additional traffic and delays in the network, and it gets worse when the failure rate of the nodes increases. The reliability of the system can be increased using multipath routing [22]. Multipath routing allows the establishment of more than one paths between source and destination and provides an easy mechanism to increase the likelihood of reliable data delivery by sending multiple copies of data along different paths. However, using this technique, the traffic volume increases significantly.

In this chapter a data splitting mechanism for wireless ad-hoc and sensor networks with a highly dynamic topology (due to mobility and failures) is presented. This mechanism enables a tradeoff between traffic volume and the reliability. As shown in Figure 3.1, the data packet is split in n subpackets ($n =$ number of disjoined paths from source to destination). If only E_n subpackets ($E_n < n$) are necessary to rebuild the original data packet (condition obtained by adding redundancy to each subpacket), then the trade-off between traffic and reliability can be controlled.

We first designed a new routing algorithm, Multipath on-Demand Routing algorithm (further referred to as MDR). MDR is an on-demand algorithm, meaning that a new path from a source to a destination is created only when a data packet has to travel between them. The algorithm provides several paths from sources to destinations.

A Reed-Solomon Code (RSC) [18] based Modified Erasure Correction code (MEC) is designed to add redundancy to the original packets. RSC codes are linear block codes, which are often denoted $RS(n, k)$ with s-bit symbols. The encoder takes k data symbols and adds $n - k$ check symbols to make an n symbol codeword. Moreover, if the position of the error is known, RSC could correct the same number of errors as the redundancy added. In Figure 3.1 on page 39, there are two ways

Computing, Wuhan, China, 2004.

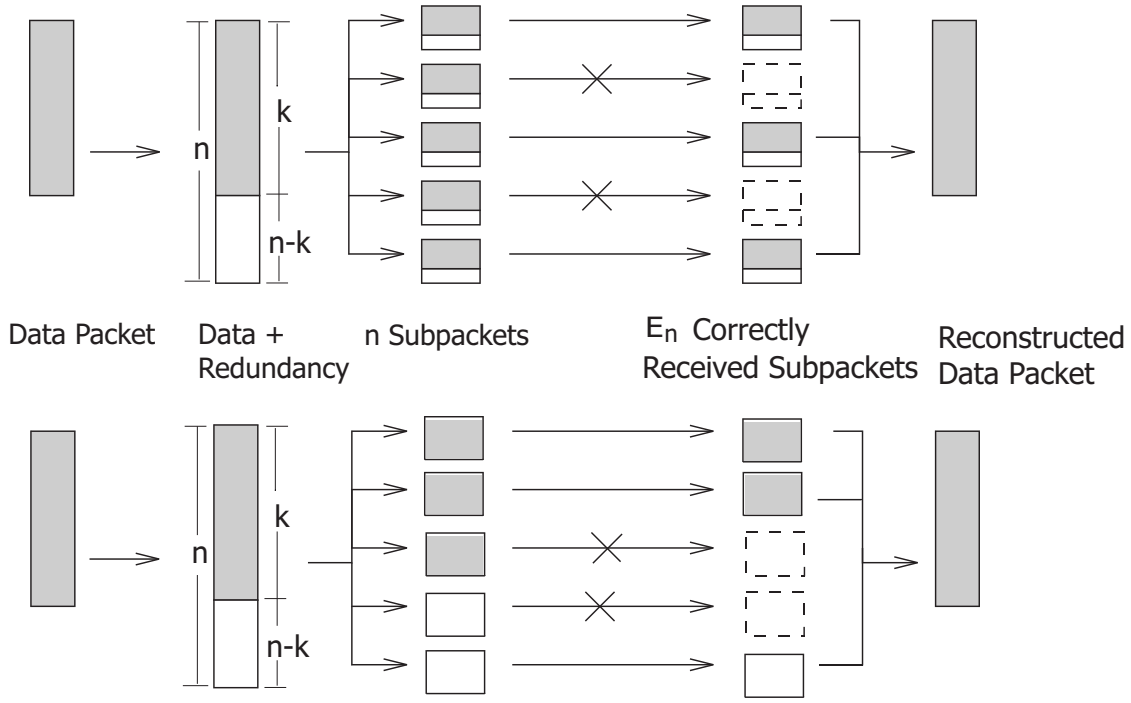


Figure 3.1: Data splitting across multiple paths

of splitting the packet and the redundancy. In our scheme, we choose to split the original packet and the redundancy data separately as shown in the lower part of Figure 3.1 on page 39. In this way, we could easily locate the error position and reconstruct the original packet with less redundance needed.

The algorithm starts by MDR discovering n multiple paths from the source to the destination. Sending the same data over all discovered paths is a solution in case of node failures but it requires large quantities of network resources (such as bandwidth and energy). Our contribution is to develop a new multipath routing algorithm that will discover several disjoint paths between the source and destination nodes. Then we compute E_n, an estimate on how many subpackets will successfully arrive to the destination, based on the failure probabilities of the paths. Then we choose a $RS(n, k)$ code with $k \leq E_n$. With this k, we split the data packets into k parts (further referred to as *subpackets*). We will make use of the *Modified Erasure Correction codes* to compute the $n - k$ redundant subpackets added to the original subpackets. Finally, we send these n subpackets instead of the whole packet, across n multiple paths. The basic principle is to transmit a sequence of n subpackets, out of which only E_n subpackets are necessary to reconstruct the original packet. The receiver's robustness to missing packets is increased, which also implies that a return feedback channel as in retransmission is not needed anymore.

Section 3.2 gives an overview of the current research in multipath routing. Then Section 3.3 presents the new multipath routing protocol. Section 3.4 focuses on how to compute E_n (an estimate on how many subpackets will successfully arrive to the destination, based on the failure probabilities of the paths) and what approximations can be made. It also explains how to generate redundant subpackets. The obtained theoretical results are verified by several simulations described in Section 3.5. The chapter ends with conclusions and directions for the future work.

3.2 Related work

This section is additional to the state of art in Chapter 2. This related work first introduces the concept and development of multipath routing. Then we give a more detailed explanation of Dynamic Source Routing, which is used in the simulation later for comparison. Finally, we make a distinction between disjoint and braided multipath routing.

3.2.1 Multipath routing

Multiple paths can be useful in improving the effective bandwidth of communication pairs, responding to congestion and bursty traffic, and increasing delivery reliability. It has been studied in several different contexts. Traditional circuit switched telephone networks used a type of multipath routing called alternate path routing to decrease the call blocking probability and increase overall network utilization. In alternate path routing, the shortest path between phone exchanges is typically one hop across the backbone network; the network core consists of a fully connected set of switches. When the shortest path for a particular source destination pair becomes unavailable, rather than blocking a connection, an alternate path, which is typically two hops, is used. Multipath routing has also been addressed in data networks which are intended to support connection-oriented service with QoS. For example, the PNNI signalling protocol has been used in Asynchronous Transfer Mode (ATM) networks to set up multiple paths between a source node and a destination node. The primary path is used until it either fails or becomes over-utilized, then alternate paths are tried.

Alternate or multipath routing has typically lent itself for use in connection-oriented networks. However, in packet-oriented networks, like the Internet, multipath routing could be used to alleviate congestion by routing packets from highly utilized links to links which are less highly utilized. The drawback of this approach is that the cost of storing extra routes at each router usually precludes the use of

multipath routing. However, multipath routing techniques have been proposed for Open Shortest Path First (OSPF) [66], a widely used Internet routing protocol.

In routing of wired networks, multipath routing has been widely developed [14] [17] [93] [101] [116]. These protocols use table-driven algorithms to compute multiple routes. Studies show, however, that proactive protocols perform poorly in mobile networks because of excessive routing overhead [11] [42]. On-demand routing protocols are widely developed for ad hoc network because they consume much less bandwidth than proactive protocols. Ad hoc On-Demand Distance Vector (AODV) [76] and Dynamic Source Routing (DSR) [43] are the two most widely studied on-demand ad hoc routing protocols. Previous work [74] has shown limitations of the two protocols. The main reason is that both of them build and rely on a unipath route for each data session. Whenever there is a link break on the active route, both routing protocols have to invoke a route discovery process. Each route discovery flood is associated with significant latency and overhead.

Multipath routing in ad hoc networks has also been proposed to increase the reliability of ad hoc networks. Several different multipath algorithms have been studied by the prior work. Intelligent multipath extensions by Napsipuri and Das [69] have been added to reduce the frequency of routing discovery flooding, while maintaining several disjoint alternate paths between source and destination. They consider the situation where the destination replies to a selected set of queries. Many duplicates of the flooded query arrive at the destination via different routes. The Queries that are replied to are those that carry a source route that is link-disjoint from the primary source route.

Lee and Gerla proposed *Split Multipath Routing* (SMR) [53], which is an on-demand multipath source routing protocol and similar to DSR. Unlike DSR, intermediate nodes do not keep a route cache, and therefore, do not reply to a route request. This is to allow the destination to receive all the routes so that it can select the maximally disjoint paths. Maximally disjoint paths have as few links or nodes in common as possible. Duplicate route requests are not necessarily discarded. Instead, intermediate nodes forward route requests that are received through a different incoming link, and whose hop count is not larger than the previously received route requests. The proposed route selection algorithm only selects two routes. However, the algorithm can be extended to select more than two routes. In the algorithm, the destination sends an route reply for the first route request it receives, which represents the shortest delay path. The destination then waits to receive more route requests. From the received route requests, the path that is maximally disjoint from the shortest delay path is selected. If more than one maximally disjoint path exists, the shortest hop path is selected. If more than one shortest hop path exists, the path whose route request was received first is selected. The destination then sends a route reply for the selected route request.

Marina and Das developed AOMDV [61], which is an extension to the AODV protocol for computing multiple loop-free and link disjoint paths. Loop-freedom is guaranteed by using a notion of *advertised hopcount*. Link-disjointness of multiple paths is achieved by using a particular property of flooding. To keep track of multiple routes, the routing entries for each destination contain a list of the next-hops along with the corresponding hop counts. All the next hops have the same sequence number. For each destination, a node maintains the advertised hop count, which is defined as the maximum hop count for all the paths. This is the hop count used for sending route advertisements of the destination. Each duplicate route advertisement received by a node defines an alternate path to the destination. To ensure loop freedom, a node only accepts an alternate path to the destination if it has a lower hop count than the advertised hop count for that destination. Because the maximum hop count is used, the advertised hop count therefore does not change for the same sequence number. When a route advertisement is received for a destination with a greater sequence number, the next-hop list and advertised hop count are reinitialized.

Zhen and Srikanth proposed another extension to AODV (AODVM) [114] for finding reliable routing paths. Intermediate nodes are not allowed to send a route reply directly to the source. Also, duplicate route request packets are not discarded by intermediate nodes. Instead, all received request packets are recorded in an route request table at the intermediate nodes. The destination sends a route reply for all the received route request packets. An intermediate node forwards a received route reply packet to the neighbor in the route request table that is along the shortest path to the source. To ensure that nodes do not participate in more than one route, whenever a node overhears one of its neighbors broadcasting a route reply packet, it deletes that neighbor from its route request table. Because a node cannot participate in more than one route, the discovered routes must be node-disjoint.

The *Temporally Ordered Routing Algorithm* [73] proposed by Park and Corson, provides loop free multiple alternate paths for mobile wireless network by maintaining a "destination oriented" directed acyclic graph (DAG) from the source. It rapidly adapts to topological changes, and has the ability to detect network partitions and erase all invalid routes within a finite time. Another candidate for multipath routing for WSN is *Directed Diffusion* [41], which features data centric dissemination and in network data aggregation. It can realize robust multipath delivery, empirically adapt to a small subset of network paths, and achieve significant energy savings when intermediate nodes aggregate responses to queries. Based on directed diffusion, a novel braided multipath routing scheme, which results in several partially disjoint paths, is studied by D.Ganesan [29]. Results show it is a viable alternative for energy efficient recovery from failures in WSN.

All of the above mentioned routing algorithms focus on using multipath routing only to reduce the effects of failures. To increase the reliability of the network, these

algorithms send multiple copies of the same data across multipath, which invokes a large amount of traffic.

3.2.2 Dynamic source routing

In Section 3.5, we compare the performance of our protocol with Dynamic Source Routing [43], which is a simple and efficient on-demand routing protocol designed specifically for use in multi-hop wireless ad hoc networks of mobile nodes. DSR involves the following phases:

- *Route Request* - the source floods the network with messages, trying to find in this way the destination. The messages increase in length by each hop they travel. If more than one route request message reach a node, only the first one is processed and the others are discarded.
- *Route Reply* - if the destination receives a route request message from the source it will reply with a message containing the path used to reach the source. In the case of bi-directional links, this path is simply reversed. The reply messages have constant length between the source and the destination. Still, their initial length depends on the number of hops between the source and the destination.
- *Route Maintenance* - after the source has received a path to the destination, it sends the data packet on it. Each node is responsible for ensuring that the message travels to the next hop (this can be done for example by passive acknowledgement [45]). If a node detects that a link is broken, it sends this information back on the path to the source. A new path has to be constructed or another cached path can be used. The length of the messages involved in this phase is dependent on the number of hops between them.

3.2.3 Disjoint and braided multipath

Out of many possible designs for multipath routing protocols, two distinct mechanisms exist: disjoint and braided.

- *disjoint multipath* routing tries to construct alternate paths which are node disjoint with the primary path, and with each other. Thus they are unaffected by failure on the primary path. But those alternate paths could potentially have much longer latency than the primary path and therefore consume significantly more energy than that on the primary path.

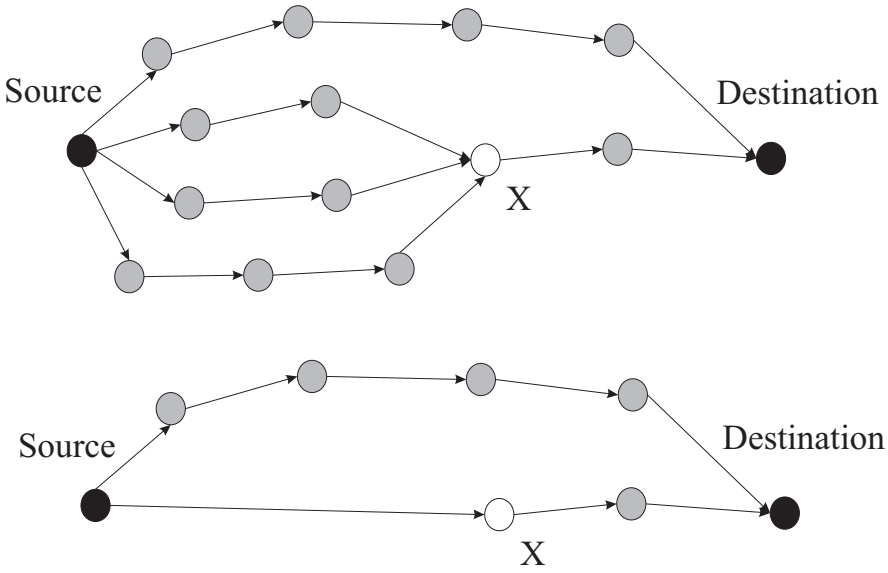


Figure 3.2: Braided multipath case

- *braided multipath* routing relaxes the requirement for node disjointness, which means alternate paths in a braid may partially overlay with the primary path and so are not completely node disjoint.

In this work, we only consider the disjoint multipath cases, because the braided multipath can easily be transformed into disjoint multipath. As shown in Figure 3.2 on page 44, there exists four multipaths between the source and destination and three of them are overlaid. The data splitting algorithm uses only the failure probability of each path. This means that we can consider the three paths from source to node X as a separate disjoint multipath problem, apply the algorithm presented in the next section to it and get a failure probability for it. This way we can reduce the group of braided paths to a single path. It is easy to see that the simple example presented here can be generalized for any type of configuration.

3.3 Multipath on-demand routing

Multipath Routing allows the establishment of multiple disjoint paths between source and destination, which provides an easy mechanism to increase the likelihood of reliable data delivery by sending multiple copies of data along different paths. Based on Dynamic Source Routing (DSR)[45], we designed a new multipath routing algorithm Multipath on-Demand Routing algorithm (further referred to as MDR). The algorithm provides several paths from the source to the destination. A data splitting

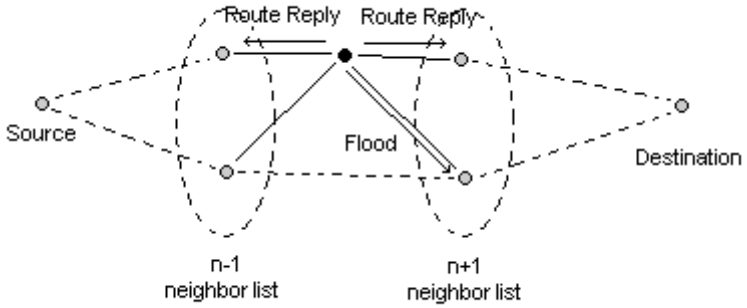


Figure 3.3: MDR algorithm details

algorithm as presented in Section 3.4 will be used to safely route data while keeping the amount of traffic low. The two phases of the algorithm are described below.

The MDR algorithm has two phases (see Figure 3.3 on page 45):

- *Route Request* - when the source wants to find a destination it floods the network with a short message announcing this. The message contains the source ID, the destination ID and the ID of the request. Thus, the length of the message remains constant during the route request.
- *Route Reply* - the destination will eventually receive one of the route request messages. It only knows that there exists a path. It is not interested in what the path is. The destination just returns a route reply to the neighbor from which it received the route request message. The message contains a supplementary field that indicates the number of hops it travelled so far. When this neighbor node receives the route reply, it increments the hop count of the reply message and then forwards the route reply to the neighbor from which it got the original route request.

This mechanism reduces the size of the messages considerably when compared to the original DSR. In fact we are moving the information stored inside the messages to the sensor nodes themselves. The sensor nodes are responsible to "remember" where the flooding message came from.

One can notice that there is no route maintenance. This approach will be discussed more in detail after the way the multiple paths are handled and the simulation results are presented.

The second group of modifications involve the multiple paths management. In the original DSR, if the same route request message was received several times by a

For source node **S**

1. **IF S** has new packet to send and no route is known to the targeted destination
2. **THEN** forward route request message to all neighbor nodes of **S**; set route discovering timer;
3. **END IF**
4. **IF S** receives route reply from destination
5. **THEN IF** this is the first route reply
6. **THEN** set reply timer;
7. **END IF**
8. **IF** reply timer is not expired
9. **THEN** record the route and wait for more route reply
10. **ELSE** start transmission of the packet
11. **END IF**
12. **END IF**
13. **IF S** does not receive route reply from destination before route discovering timer expires
14. **THEN** restart route request (go to 1)
15. **END IF**

Figure 3.4: Pseudo code of the source node in MDR

node, only the first one was considered and the rest were discarded. MDR considers all the messages and uses the whole information it can get out of them.

By using these changes we obtained a controlled flooding in the first phase of the algorithm by using small messages with fixed length. The second phase also uses small fixed length messages that involve only a fraction of nodes existent between the source and the destination.

The following parts gives detailed explanation of the algorithm. The pseudo code of the algorithm is shown in Figure 3.4 on page 46, Figure 3.5 on page 48 and Figure 3.6 on page 49.

3.3.1 Route request phase

The Route Request phase is the mechanism by which the source of the data packet notifies the destination that it has a packet for it. The route request message involves all the nodes of the network or at least, only the nodes to which the message arrives before it expires (the message contains a field saying how many hops the message is allowed to travel).

Message description

The route request message contains the following fields:

- *snodeID* the source node ID
- *dnodeID* the destination node ID
- *floodID* the route request message ID
- *lasthop* the ID of the node forwarding this message
- *ack* the ID of the last hop

For the algorithm to work, each node in the network has to have an unique ID. Each message source maintains a counter of the requests sent, such that each route request message in the network is uniquely identified by the first three fields. The *ack* field is needed to distinguish between the messages received by a node. This way, a route request message can be immediately classified as being received for the first time, or being just a passive acknowledgement of a previously sent message.

Route Request phase description

When a source node has to transmit a message to a destination, it first checks its cache to see if there are any routes to that destination that did not expire. If not, it generates a new route request message filling the ack field with its own ID.

When receiving a route request message, a node checks its local data structure to see if it has received another route request message having the same three fields identical. If not, it creates a new entry in the data structure and stores this information plus the ID of the node from which it received it. From additional messages received the node has to store only the name of the neighbor. It can easily check and mark if the source of the message is a first order neighbor by looking at the lasthop or ack fields.

The node will forward only the first route request message it gets. It has to change only the ack field with the lasthop value and the lasthop with its own ID. After receiving several such messages each node knows who are its neighbors and more than that which ones are closer to the source (further referred as the *n-1* neighbor list) and which one closer to the destination (further referred as the *n+1* neighbor list). In fact, each data structure stores them in two separate lists according to the previous rule. If the node identifies itself as being the destination of the message it initiates the second phase of the algorithm (the route reply phase).

For intermediate node **i**

```
1.  IF node i receives route request message from neighbor j
2.    THEN IF same route request is not received before
3.              THEN remember the S node, D node, ID of the request and the
                      neighbor j; forward the route request with new ack and
                      lasthop; put neighbor j in n-1 neighbor list
4.           END IF
5.           IF same route request is received from neighbor j before
6.            THEN discard the route request
7.           ELSE IF ack field in the route request is i
8.                    THEN put neighbor j in n+1 neighbor list
9.                   ELSE put neighbor j in n-1 neighbor list
10.                  END IF
11.          END IF
12. END IF
13. IF node i receives route reply message from neighbor j
14.   THEN IF node i is addressed by the route reply and this route
15.              reply has not been forwarded by node i before
16.            THEN IF n-1 neighbor list is not empty
17.                    THEN forward the route reply to the first neighbor
                              in n-1 neighbor list
18.                  ELSE IF detours field in route request is larger
                              than 0 and n+1 neighbor list is not empty
19.                          THEN forward the route reply to the first
                                  neighbor in n+1 neighbor list
20.                         ELSE discard the route reply
21.                         END IF
22.                  END IF
23.           ELSE IF same route reply is received before
24.                    THEN remove neighbor j form both n-1 and n+1
                              neighbor list
25.                 END IF
26.          END IF
27. END IF
```

Figure 3.5: Pseudo code of the intermediate node in MDR

For destination node **D**

1. **IF D** receives a route request addressed to it from neighbor **j**
2. **THEN IF** same route request is not received before
3. **THEN** remember node **S**, ID of the request and the neighbor **j**; set timer
4. **END IF**
5. **IF** timer is not expired
6. **THEN** send route reply to neighbor **j**
7. **END IF**
8. **END IF**

Figure 3.6: Pseudo code of the destination node in MDR

3.3.2 Route reply phase

The Route Reply phase is the part of the algorithm in which several paths between the destination and the source are reported to the source (if they exist). The reply messages have fixed length. Because in the previous phase each node stored information about the neighbors that forwarded the route request message, the complete path between the source and the destination does not have to be stored inside the message.

Message description

The route reply message contains two groups of fields. The first group of fields uniquely identify the instance of the route reply, which consists of the following fields:

- *snodeID* the source node ID
- *dnodeID* the destination node ID
- *floodID* the flood message ID

The second group of fields changes when each intermediate node forwards the route reply, which consists of the following fields:

- *lasthop* the ID of the node forwarding this message
- *nexthop* the ID of the node to which the message is forwarded
- *ack* the ID of the last hop

- *hops* the number of the hops the message travelled through
- *detours* the number of detours a message can take

The meaning of the field names is the same as in the previous phase. There are two new fields: the *nexthop* field contains the ID of the node that has to receive this message. This information is provided by each node from their local data structure. The *hops* field is incremented with each hop the message travels and represents the current path length. The *detours* field specifies how many times the reply message is allowed to travel in an opposite direction (from source to destination).

Route Reply phase description

When the first route reply message arrives at the source, this node stores the ID of the node that forwarded the message and the path length. It also sets up a timer to measure the interval that it will wait for other reply messages to come. When this timer expires it splits the original data message according to the number of paths, the maximum probability of failure and the length of the paths and forwards it. The paths can also be stored in a local cache (together with time information) for future usage (this feature is not implemented yet).

A node that receives a route reply addressed to it, will modify the second group of fields in the message according to the new parameters. Afterwards, it will forward the modified route reply to the first neighbor in the *n-1* neighbor list. If this list is empty and the detours field is not empty, it chooses the first neighbor in the *n+1* neighbor list and also decreases the *detour* variable by 1. A node that receives a route reply not addressed to it, searches its own data structure to find the entry corresponding to the first three fields. If such an entry is found, it removes the forwarding node from both *n-1* and *n+1* neighbor lists.

A node that forwarded a message has to take care of two more things: first it sets a flag in his data structure saying that it will not forward any other message and second, it waits for the passive acknowledgement. If this does not arrive it assumes that the node to which it send the message is no longer there, is broken or it forwarded a message previously and it deletes it from his lists. It will try to resend the message to the next neighbor in the lists, until the lists become empty or the *detour* field becomes 0.

The previous step of removing nodes from the list is needed to ensure that the source will receive only disjoint paths. If for various reasons, the paths from the destination to the source have to be known, each node that forwards a route reply message can append its ID to it. This way, the messages will grow in length, but this growth is controlled and involves only a subset of the nodes.

After route request and route reply phase, the source obtains n multiple par-

allel paths to the destination node. When a link failure appears in the path, the intermediate node sends a route error message back to the source node. When the number of healthy paths is smaller than the estimated number of successfully paths, the source node will reinitiate a route request.

3.4 Data splitting across multiple paths

The route discovery process of MDR provides multiple disjoint paths between source and destination. In this section we try to predict the number of paths that will succeed in delivering subpackets. Furthermore, we will give an approximation that allows for a term that can be used to increase the chance of successful delivery of the entire message at the trade-off of added redundancy.

Of course, one can send the whole message along each of the available paths, but the overhead induced by this will be too high. The entire data packet to be sent from the source to the destination over the available n disjoint paths will be split up into smaller subpackets of equal size with added redundancy. The number of created subpackets corresponds to the number of available paths. Only a smaller number of these subpackets will then be needed at the destination to reconstruct the original message. In the following, we will focus on approximating with high probability the number of successful paths E_n that gives n multiple paths. This value will then be used to determine the parameter k in $RS(n, k)$ of *Modified Erasure Correction codes*. $k(k < n)$ is the number of subpackets the original packet should be splitted into. And then we could add $n - k$ redundancy subpackets for the split message transmission as in Section 3.4.2. The total number of subpackets n as well as the added redundancy $n - k$ is dependent on the multipath degree and on the failure probabilities of the available paths. As these values change according to the positions of the source and the destination in the network, each source must be able to decide on the parameters for the error correcting codes before the transmission of the actual data subpackets.

3.4.1 Expected number of successful paths (n given)

Suppose we want to send a data package from a source to a destination and the process of route construction is finished, resulting in n disjoint paths that are to be used. To each path has some rate $p_i (i = 1, \ldots, n)$ is associated that corresponds to the probability of successfully delivering a message to the destination.

This setting corresponds to a repeated Bernoulli experiment, the i-th subrun corresponding to the message transmission along the i-th path. Note that we con-

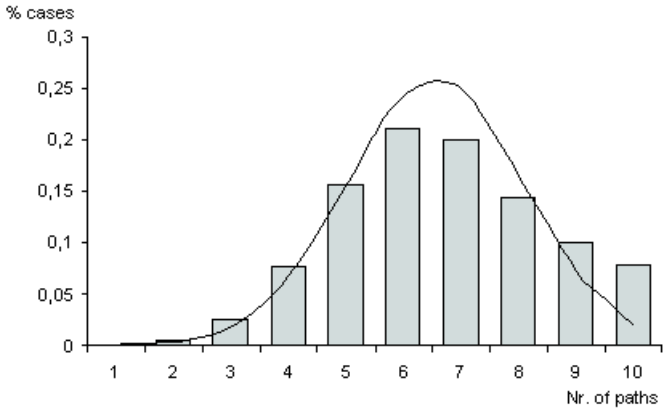


Figure 3.7: Normal distribution graph for the estimated μ and σ

sider node-disjoint paths and therefore these experiments are independent of each other. Let $S_n : \{0,1\}^n \to \mathbb{N}$ be the variable corresponding to the number of successfully delivering paths. (For S_n, each subrun is assigned a 1 if the transmission was a success along the respective path, and 0 if it failed. Then, S_n represents the sum of these for the subruns, and clearly $S_n \leq n$.) Then, the expectation for the total number of successful paths is given by

$$E(S_n) = \sum_{i=1}^{n} p_i. \qquad (3-1)$$

The distribution of the above repeated Bernoulli experiment can be approximated by a normal distribution. The accuracy of the approximation is increased with the number of experiments. However, for a small n, a suitable continuity correction could be used to obtain a better approximation. This approximation will be used to obtain a good estimator for E_n for a given bound α, the overall probability of successfully reconstructing the original message at the destination. More formal, we want to deliver a good estimation for the value of E_n for a desired bound α such that

$$P(S_n \geq E_n) \geq \alpha \qquad (3-2)$$

holds.

In order to approximate the Bernoulli experiment by normal distribution $N(\mu, \sigma)$, the mean μ and the standard deviation σ are needed. In our case, μ will be given by the expectation for S_n, i.e. by the sum of the probabilities of successful delivery along each path, and thus we set

$$\mu := E(S_n) = \sum_{i=1}^{n} p_i. \qquad (3-3)$$

α	95%	90%	85%	80%	50%
x_α	-1.65	-1.28	-1.03	-0.85	0

Table 3.1: Some values for the bound α

Accordingly, we obtain the standard deviation by setting

$$\sigma^2 := \sum_{i=1}^{n} p_i(1 - p_i). \qquad (3-4)$$

Figure 3.7 on page 52 shows as an example for each number of successful paths, the percentage of the test cases. It is a 100,000 run histogram for a multipath of degree 10, each path having 5 to 15 intermediate nodes between source and destination (average is 10), and each node having a failure probability of 0.2 for 20% of the simulation time. This corresponds to an average probability of a successful delivery of $p_i = 0.96$ for each node. Additionally, the graph for the normal distribution with the estimated values of $\mu = 6.65$ and $\sigma = 1.5$ is given.

Obviously, each combination of the multipath degree n and different probabilities $p_1, \ldots, p_n$ will yield a different normal distribution. To overcome this problem, we transform to the standard normal distribution $N(0, 1)$. The variable

$$S_n^* := \frac{S_n - \mu}{\sigma} \qquad (3-5)$$

is $N(0, 1)$-distributed.

Now, consider a given bound α for the desired probability of being able to reconstruct the original message at the destination after being sent along the different paths. For the standard normal distribution, the values of the bound x_α for any given α such that the probability

$$P(S_n^* \geq x_\alpha) \geq \alpha \qquad (3-6)$$

holds are known. Some value-pairs are presented in Table 3.1 on page 53. Note that these values are independent of the number of paths n used to send data.

Using the above estimations, we transform the argument

$$S_n^* = \frac{S_n - \mu}{\sigma} \geq x_\alpha \qquad (3-7)$$

and obtain the following probability

$$P(S_n \geq x_\alpha \cdot \sigma + \mu) \geq \alpha. \qquad (3-8)$$

We can therefore use $x_\alpha \cdot \sigma + \mu$ as E_n for the forward error correction code and set

$$E_n := \max\{\lfloor x_\alpha \cdot \sigma + \mu \rfloor, 1\}. \qquad (3-9)$$

In terms of the input for the decision algorithm, the value for E_n is given by the following expression

$$E_n = \max\{\lfloor x_\alpha \cdot \sqrt{\sum_{i=1}^{n} p_i(1-p_i)} + \sum_{i=1}^{n} p_i \rfloor, 1\} \qquad (3-10)$$

which gives the number of successfully delivering paths with the overall success probability of α.

After MDR discovers n multiple paths from the source to the destination, we can compute E_n based on the failure probabilities of the paths with the given formula (3-10). Then we choose a $RS(n,k)$ code with $k \leq E_n$ to split the data packets into k parts. We will make use of the *Modified Erasure Correction codes* in the next section to compute the $n-k$ redundance subpackets added to the original subpackets. Finally we send these n subpackets across n multiple paths. Then the destination can reconstruct the original packet with a success probability higher than α.

3.4.2 Modified erasure correction

The design of error correction meets the requirements of our split-multipath scheme. The Modified Erasure Correction code (MEC) described in this section is based on the well know Reed-Solomon error correction code (RSC) [18].

RSC are linear block codes, which are often denoted $RS(n,k)$ with s-bit symbols. The encoder takes k data symbols and adds check symbols to make an n symbol codeword. RSC correct up to t errors in a codeword where $2t = n-k$. For a symbol size s, the maximum codeword length (n) is $n = 2^s - 1$. Because RSC corrects symbol errors, they can potentially correct many bit errors. This makes RSC very good at correcting large clusters of errors. Moreover if the position of the error is known, then the decoding procedures can correct up to $2t$ erasures(errors). It means that RSC could correct the same number of errors as the erasure added.

In our scheme, we choose to split the original packet and the redundancy data separately as shown in the lower part of Figure 3.1 on page 39. In this way, we could easily locate the error position and reconstruct the original packet with less redundance. In MEC, when a data packet arrives, it is divided into k subpackets each with L bits. Then these subpackets are put into a two-dimensional array of

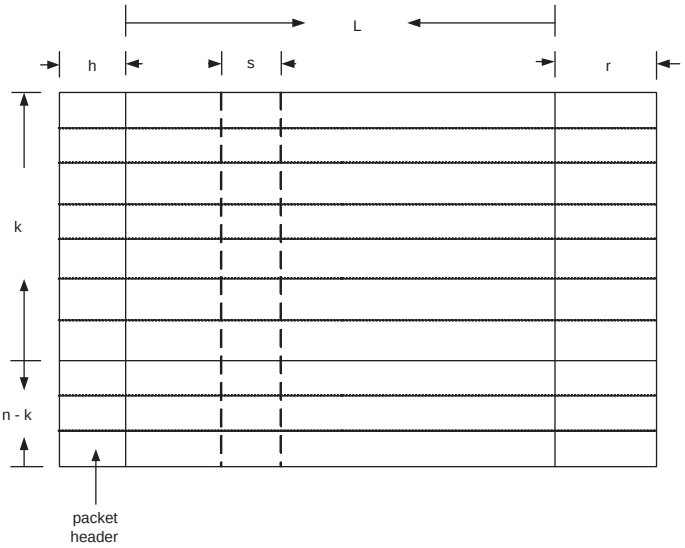


Figure 3.8: Modified Erasure Correction Code

$k \times L$ bits as shown in Figure 3.8 on page 55. Further let $L = L' \times s$ and every s bits form a symbol in finite field GF(2^s). The encoding can be carried out in two stages. The outer-codes are Reed-Solomon codes over GF(2^s) which protect against subpacket loses. Each column of information symbols in GF(2^s) is encoded into a code word of $C_0(n, k)$, where the number of redundant symbols $R = n - k$. In total there are L' outer code words in this array. Then a header h is added to each row, which keep the index of each subpacket and the number of padding added. The inter encoding is optional which gives extra reliability over link errors. A binary BCH code could be used for each row as an inner correction code. After the MEC encoding, each row of subpacket is sent on n different path established by the multipath routing algorithm. As along as more than k of them are received in the destination node the MEC is able to reconstruct the original packet.

The desired characteristics of the MEC are summarized below:

- BURST CORRECTION: Errors occur because of link failures. Normally the whole subpacket is lost instead of bit errors.
- ERASURE CORRECTION: The index in the subpacket header help to locate the error in the decoding, which results in erasures,
- ADAPTABILITY: The number of multipaths and link quality channel varies over a wide range in a short period of time. MEC can adapt to the changes quickly.

3.4.3 An example

Here we summarize how the algorithm works with an example as shown in Figure 3.9 on page 56. Our algorithm starts by MDR discovering $n = 5$ multiple paths from the source to the destination. Then we use the formula (3-10) to compute the estimate number of successful path $E_n = 3$ based on the failure probabilities of the paths. Then we choose a $RS(n, k)$ code with $k = 3 \leq E_n$. With this $k = 3$, we split the data packets into *3* subpackets. Then we make use of the MEC to compute the $n - k = 2$ redundance subpackets added to the original subpackets. Finally, we send these $n = 5$ subpackets instead of the whole packet, across *five* multiple paths. As long as at least three subpackets are successfully received, we can reconstruct the original data packet.

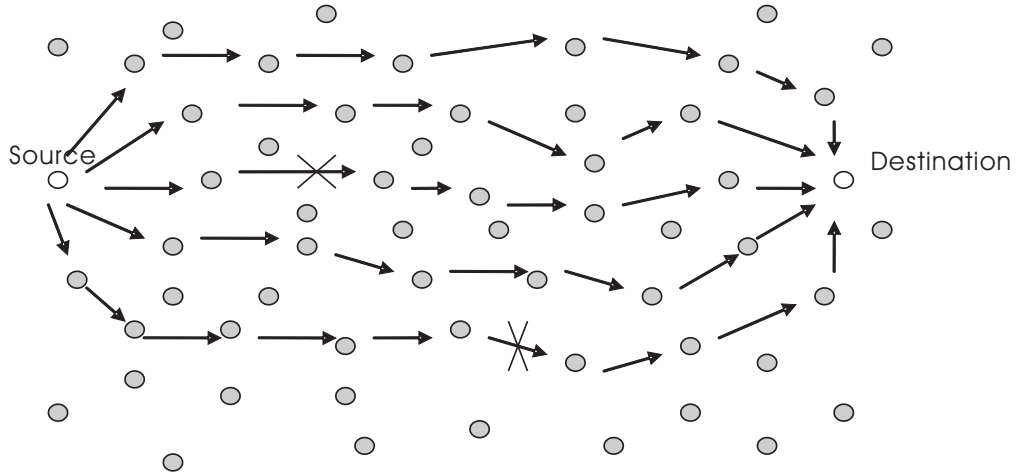


Figure 3.9: An example of algorithm with n = 5, $E_n = 3$ and k = 3

3.5 Simulation and results

We have implemented the simulation of the MDR algorithm and tried to quantify the amount of overhead it introduces versus the improvements obtained.

3.5.1 Simulation environment

The simulations were performed using a simulation template for wireless mobile sensor networks [85]. It is built upon the OMNeT++ networks simulator [100] and it contains the main features needed for simulating large ad-hoc networks of autonomous mobile nodes with sensing capabilities.

The OMNeT++ simulator is a discrete event simulator originally designed for fixed, wired, distributed systems (such as: computer networks, multiprocessor systems, etc.). The simulated objects communicate with each other by exchanging messages at discrete moments of time. In OMNeT++, simulated objects are represented by modules. The modules can be simple or composed (depth of module nesting is not limited). The modules communicate by messages (sent directly or via gates). Each module description consists of: an interface description and a behavior description. All the simulated objects (modules, gates and links) can be created either statically (at the beginning of the simulation, from the configuration files), or dynamically (during the simulation).

The wireless sensor networks are a particular example of autonomous decentralized systems. In the European EYES project (IST-2001-34734) on self-organizing and collaborative energy-efficient sensor networks, a simulation template was designed because the OMNeT++ simulator does not include support for mobile networks that communicate wirelessly. We designed this framework in such a way that allows easy modifications of the main parameters and, at the same time, the implementation details are transparent to the user. The template consists of a network of mobile nodes that can communicate by wireless means. Each node is responsible for defining its own trajectory and announcing it to the simulator. The nodes exchange messages using wireless communication. A message will be heard by all the neighbors situated within the transmission range (the modules within transmission range are connected automatically to each other). Moreover, the user can specify whether unidirectional or bidirectional links have to be used. Each node can specify and update its transmission range independently. Some basic energy management is also included in the template. The nodes have different kinds of failure probabilities (individual failures, failures that affect regions of the map, etc.). Maps for failures can be specified and used.

As shown in Figure 3.10 on page 58, the template also provides a graphical interface useful for debugging or illustration purposes and a collection of tools that help modifying or implementing new features in an easy fashion.

3.5.2 Comparison with the DSR algorithm

We have considered 50 nodes randomly distributed inside a rectangle surface (500 by 800 units). For the movement of nodes we used the Random Way Point model [23], which is a commonly used synthetic model for mobility in wireless ad hoc networks. The sleep time of a node was randomly chosen between 1 and 10 seconds and the average sleep time is 5.5 seconds. We are assuming that all the links in the network are bidirectional. Ten simulations were performed for each different combinations

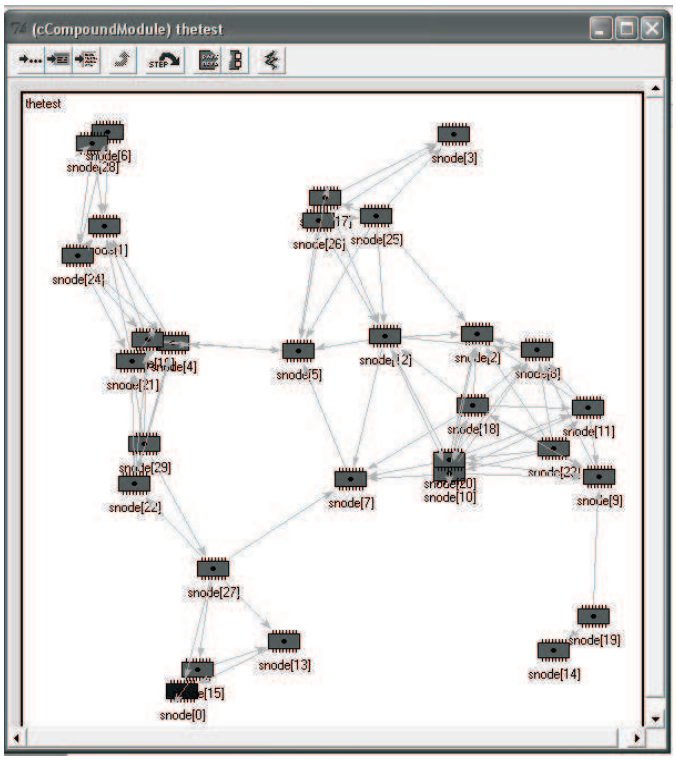


Figure 3.10: Graphical interface of simulation template in OMNeT++

of the average speed of nodes and transmission ranges. Usually, the speeds were considered in the interval 2 - 20 units/second to simulate a pedestrian speed. The transmission range is selected to be in the interval 100 - 325 units to simulate a typical sensor transmission range. One of the nodes defined as being the source node, and randomly chose a destination each 0.5 seconds to forward a message to it. Each simulation had a limit of 200 seconds (in fact after 200 seconds the source stopped generating requests and the simulations ran until all the messages were exhausted in the network).

The main parameters considered were the number of messages, the amount of traffic generated, the latency introduced and the connectivity of the network. An implementation of DSR with caching of the paths and the route maintenance enabled was also implemented for comparison. We have run both DSR and MDR for several network configurations. The parameters were identical for both cases and also the generation of destinations. The DSR algorithm had the caching of the paths and the route maintenance enabled. The results are presented in Figure 3.11 on page 59, which merely indicates the trends between the two compared the protocols.

The figure shows that the number of overhead messages is higher for the MDR

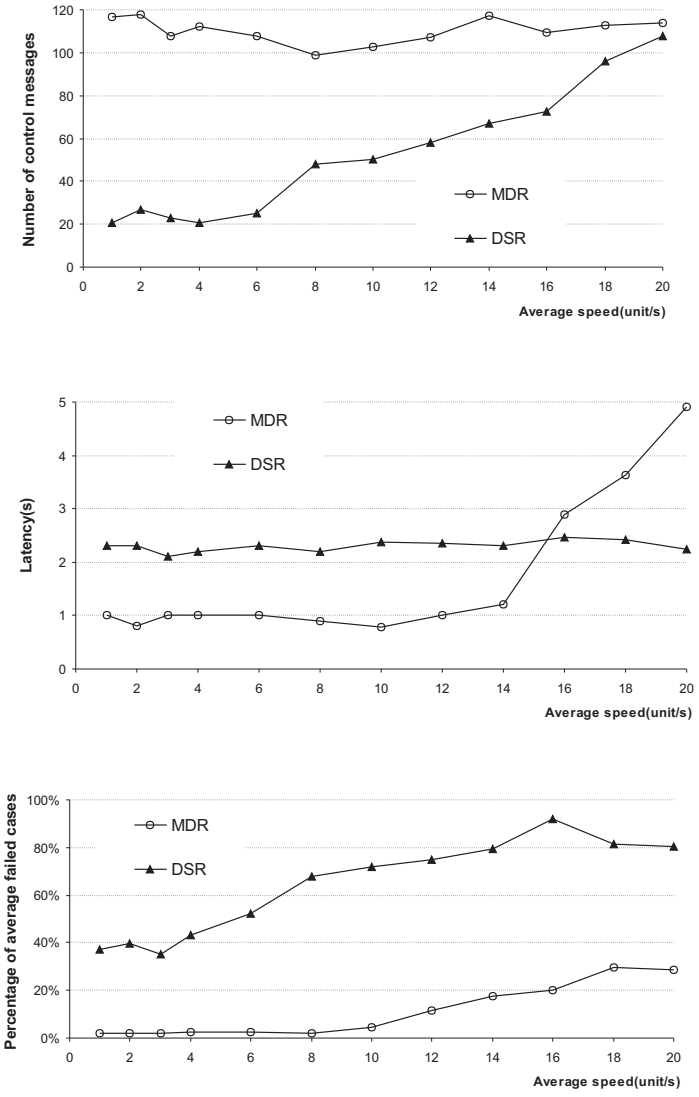


Figure 3.11: Comparison between MDR and DSR

algorithm. A closer look at the message sizes shows that the MDR traffic compared to the DSR traffic varies from a 4.04:1 to a 1.02:1 ratio (from the lower average speed to the higher one).

After the paths are created, the source will deliver one data packet that takes one round to travel through one hop. In this case, the latency of DSR is smaller with low mobility. With the increase of speed, the situation changes. In practice we assume data packets far larger than control messages. The last graph in Figure 3.11 on page 59 shows the average number of failed cases for the two algorithms. The MDR algorithm performs way better than DSR. The figure shows clearly the two objectives of our algorithm: it improves the reliability and it makes the network almost immune to higher average speed of the nodes.

A failed case is the situation in which the source had a data message to deliver to the destination but failed reaching it. There are two reasons for it:

- the route discovery mechanism did not return any valid paths between the source and the destination;
- although there were several paths available, the data packets got lost on the way (due to mobility issues).

The MDR algorithm performs way better than DSR, so this is the advantage for which we pay with a higher number of control messages and higher latency. However, we still need to consider the traffic caused by the amount of redundant data packet added by the multipath. The trade-off between reliability and traffic is analyzed and simulated in Section 3.5.6.

3.5.3 Influence of speed

This set of experiments took into consideration different degrees of mobility and network density. We put 30 to 60 nodes with transmission range of 150 units randomly distributed inside a rectangle surface (600 by 800 units).

The metrics taken into consideration were: number of control messages exchanged, number of paths discovered, latency, number of times when the multipath algorithm failed and number of times no data packet was received. For a given network setup we have varied the average speed of the nodes and noticed how these parameters varied. All the experiments were repeated for different densities of the network.

Number of control messages

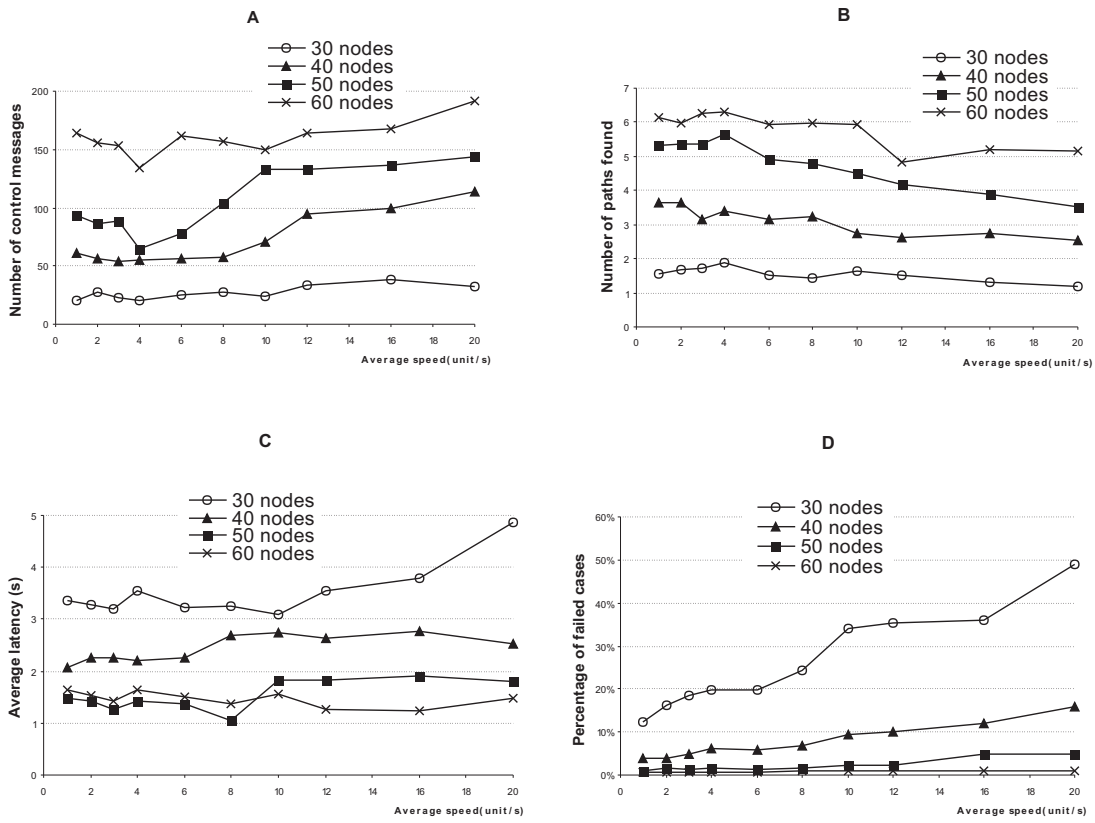


Figure 3.12: Protocol performance under different network mobility and density

The curves in Figure 3.12-A on page 61 represent the total number of control messages divided by the number of cases when at least one data packet was delivered between the source and the destination. The influence of mobility on the number of control messages is not significant. It increases slightly with the increase of mobility. However, the figure shows that the network density plays a more important role. As the density increases, the amount of control messages grows accordingly. This is largely due to the multipath degree grows with the network density, so that more control messages need to support more paths.

Multipath degree

Figure 3.12-B on page 61 presents the average number of paths discovered between the source and the destination. It is related to the average connectivity of the nodes and to their average speed. As expected, it decreases with the speed and increases with the network density. It is interesting to notice that when the network density was small (30 nodes), the network was quite often partitioned. This is why

the average number of discovered paths is only a little more than 1.

Latency

Figure 3.12-C on page 61 shows the latency introduced by the MDR algorithm. As in the previous graphs, the speed does not have significant influence on the average latency. However, the latency appears to decrease with the increase of network density. When the network density is high, it is more likely that the source node finds a path and even more finds a shorter path to the destination. And when the network density is low, the network is more partitioned and data message experiences more delay in the forwarding.

Data delivery failure cases

Figure 3.12-D on page 61 shows the percentage of data delivery failure case. This assumes that there was at least one path found from the source to the destination, and it (or all of them) failed during the data transmission. As one can see, there are very poor results for low network density. This is in principle due to the cases when the network was partitioned. Moreover, the speed influence can be reduced by increasing the network density or the average connectivity. For higher connectivity values, the speed has almost no influence at all. This is a main difference between MDR and DSR. For higher mobility, DSR becomes almost unusable while MDR still performs better.

3.5.4 Waiting time tuning

After receiving multiple route replies messages, the source can decide how to split the data packet and forward it in order to obtain maximum reliability and lower amount of traffic. There is a certain time interval in which the source waits and stores the route replies (further referred to as the *waiting interval*).

The *waiting interval* is a key factor for the algorithm. Considering an ideal network in which the nodes do not fail in any way, there is an optimum value for this *waiting interval*. When using a shorter interval, additional routes discovered are discarded. When using a bigger one, the probability for routes to expire increases. Figure 3.13 on page 63 and Figure 3.14 on page 64 present the results of varying the *waiting interval* for different average speed value. The graphs show the number of paths discovered and the total number of failed cases (there were no replies received at the source from the destination). The case of average speed 15 illustrates better the previous phenomenon that longer or shorter waiting interval can increase the failures.

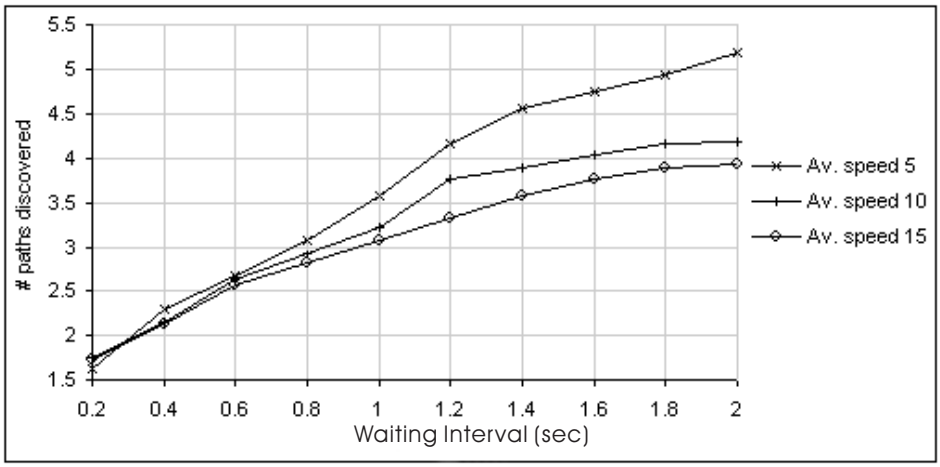


Figure 3.13: Number of paths discovered

3.5.5 Failures

The algorithm we presented was designed having in mind the following reasoning: multipath routing should provide superior reliability to single path routing because it uses several additional paths to deliver the data from the source to the destination. During this process, even if some of the paths fail, the data packet can still be reconstructed at the destination if enough redundancy was added to each subpacket. No acknowledgements are necessary. In a data splitting mechanism the overall latency decreases with the increase of the data packet length (this being paid back by the larger communication and processing overhead).

For our simulations we have considered only the case of temporary communication failures. Analyzing the results of the simulations we see that not only the data packets are affected by errors, but also the route discovery messages and also the passive acknowledgements. When integrating the failures into the simulation, the algorithm is still superior in performance to DSR, but there is quite a difference between this and the ideal situation presented above. The failures on route discovery messages only decrease the number of discovered paths. As the simulation results show, in a dense network these failures do not have significant effects. Because of the high connectivity of the node, the number of discovered path is also high.

Still, we introduce two measures to improve the reliability of the route discovery messages. The first method that comes to mind to combat the effects of failures is sending each control message several times (2 times in our case). When we applied this to the route request messages, we have obtained certain improvements (up to 8.57% increase in successful data delivery). When it comes to repeating the route reply messages, the effect of failure is almost negligible. An interesting effect appears.

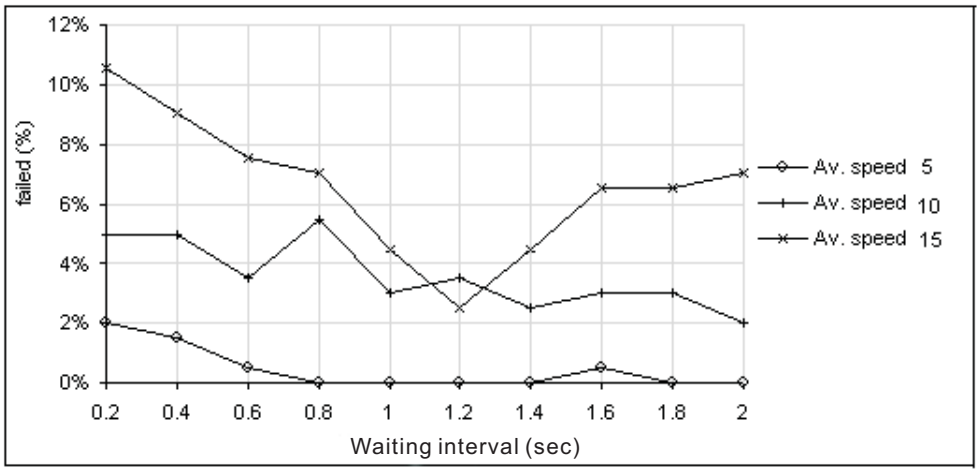


Figure 3.14: Algorithm failed cases

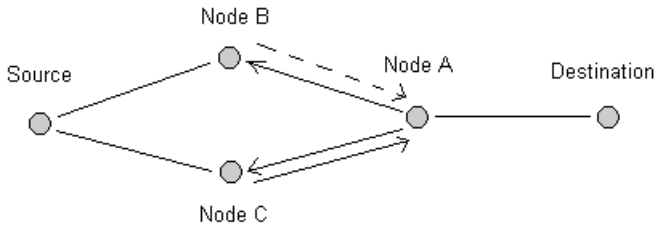


Figure 3.15: Node A fails getting a passive acknowledgement

Assume the configuration in Figure 3.15 on page 64.

Suppose that the route request phase took place and right now we have the route reply phase. Node A selects its first *n-1* neighbor (Node B) and sends it the route reply packet. Node B performs his function but for certain reasons Node A does not get the passive acknowledgement. It assumes that the link is broken and continues with the next neighbor on the list. In the end, the source receives two paths to the destination. If there are other intermediate nodes between the source and Node B or Node C, the source cannot determine from the route reply packet that in fact the two paths are not disjoint.

In this case, retransmitting messages does not solve anything. Even if Node A retransmits the message for Node B, the second one just ignores it (according to the algorithm). We could modify the algorithm to introduce some sort of Acknowledgement mechanism, but if this one fails as well we still end with braided multipaths.

The solution lies in modifying the Route Reply Message. Each node on the

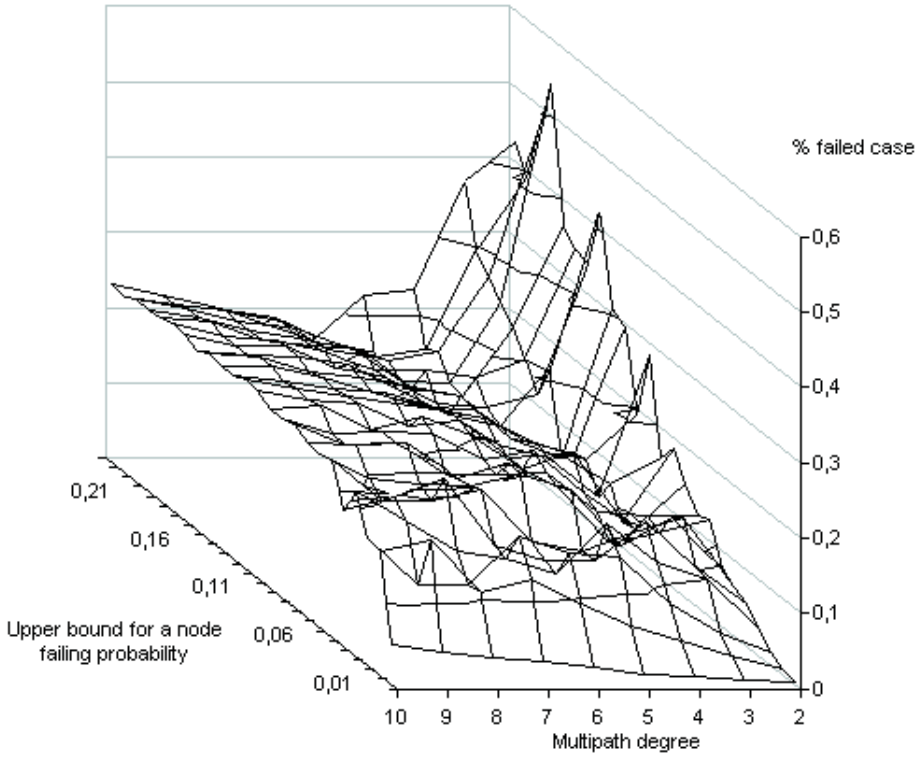


Figure 3.16: Failed communication percentage

return path should add its own ID to the message and give in this way the possibility to the destination to remove the braided paths.

3.5.6 Data splitting scheme

We have performed several simulations in order to verify the theoretical results and to get a better understanding on how well our solution works.

We have assumed that in a given sensor network there exists a routing algorithm that returns several paths from a source to a destination. We have also assumed that the multipath degree varies between 2 and 10 and each path has a length of 5 to 15 intermediate nodes (6 to 16 hops) between the source and the destination. For each simulation we considered that the sensor nodes have a failure probability P $\in [0.01, upperbound]$. This bound took values from 0.01 up to 0.25 with a step of 0.01. For each multipath degree and node failure probability bound we have performed 100000 simulations.

By a *failed communication* we understand the case when the destination could

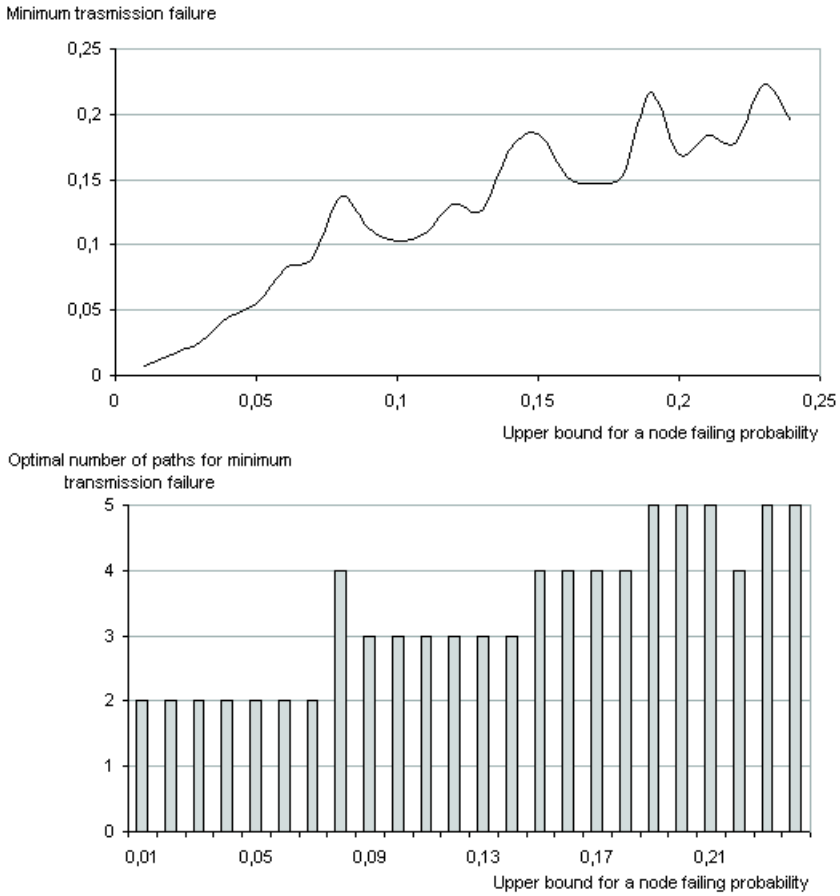


Figure 3.17: Minimum transmission failure and number of paths used

not get the original message from the source (by receiving a smaller number of subpackets than E_n). Figure 3.16 on page 65 presents the total failure probability of the algorithm as a function of the upper bound of the node failure probability and the multipath degree. In other words, this is the percentage of the failed transmission out of all the transmissions. E_n can be calculated as shown in formula (3-10) of Section 3.4 or in an approximate way in order to reduce the amount of data needed in the computation. The difference between the cases when E_n is computed in an exact way and when the approximation is used is very small - the relative error is less than 5%. The approximation is:

$$p_i = \prod_{j=1}^{n}(1 - q_{ij}) \approx (1 - \bar{q_i})^n \qquad (3-11)$$

where $\bar{q_i} = \frac{\sum_{j=1}^{n} q_{ij}}{n}$. q_{ij} represents the failure probability of the j-th node from the i-th path, while p_i is the probability of the i-th path to be successful. In the method,

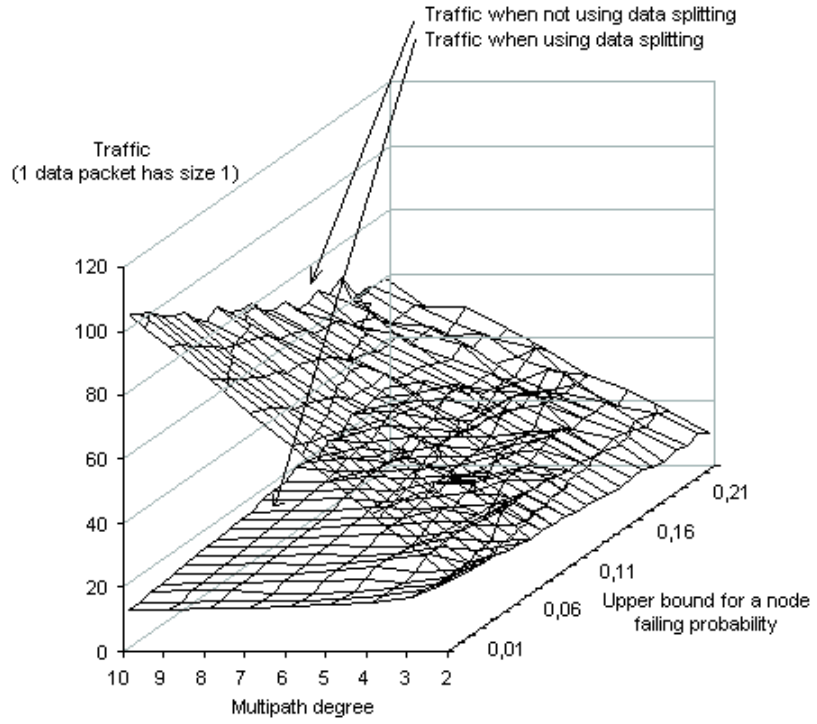


Figure 3.18: Traffic when using the data splitting and when not

we can obtain empirically the average failure probability of the nodes in the network and only use it in the calculation.

As one can notice in Figure 3.16 on page 65, for each upper bound of node failure probability there is an optimal number of multipath for which the failed transmission probability gets to a minimum. With the increase in the number of paths, the probability of error also increases! This is explained by the fact that E_n is not constant, it also varies with the multipath degree. The upper graph in Figure 3.17 on page 66 presents the minimum transmission failure that occurred in this simulations for each upper bound of node failure. While the lower one in the graph gives the corresponding number of multipaths where this minimum was achieved.

The next set of simulation results concerns the traffic used. The packet that the source needs to transmit to the destination across the multipath is considered to have the size 1 and ten transmissions have been performed for each setup. The total traffic (the case when the full data packet is sent through each subpath) and the traffic used when applying data splitting across the multiple paths is given in Figure 3.18 on page 67. The big difference between the two cases is the reason for which this algorithm should be employed: we reduce the used traffic significantly.

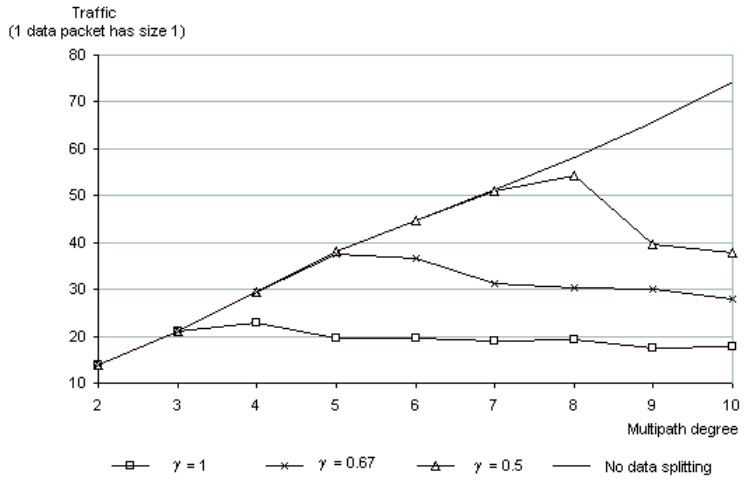


Figure 3.19: Traffic for different γ values

Finally, we have analyzed what happens when changing the x_α coefficient. Actually we use a coefficient γ such that: $E_n = \gamma \cdot E_{k0}$, where $E_{k0} = \sum_{i=1}^{k} p_i$. The variation of x_α maps on the variation of γ. Figure 3.19 on page 68 and Figure 3.20 on page 69 show how the traffic and the number of failed transmissions vary for different values of the γ coefficient (in this example we have chosen a given upper bound of the failure probability of the nodes of 0.15). For all the other upper bounds of node failure probability the values respect the same proportions. It shows a trade-off that the decrease of x_α result in a higher traffic volume in the network, while at the same time a lower failure percentage of data delivery.

3.6 Conclusions

This chapter introduced a splitted multipath scheme to control the trade-off between traffic and reliability of data routing in wireless sensor networks. An on-demand multipath routing algorithm offers the data source with several paths to any destination (if available). It is used in combination with a data splitting method based on Modified Erasure Coding.

By splitting the data across multiple paths, the traffic volume goes to much lower values compared with sending the same data across multiple path. The trade-off is the reliability of the delivered packets. While this gives us a way to adjust the reliability while keeping the data traffic low. When using a lower value for E_n than the calculated one, the traffic increases but the percentage of failures decreases (down to the case of all paths failure).

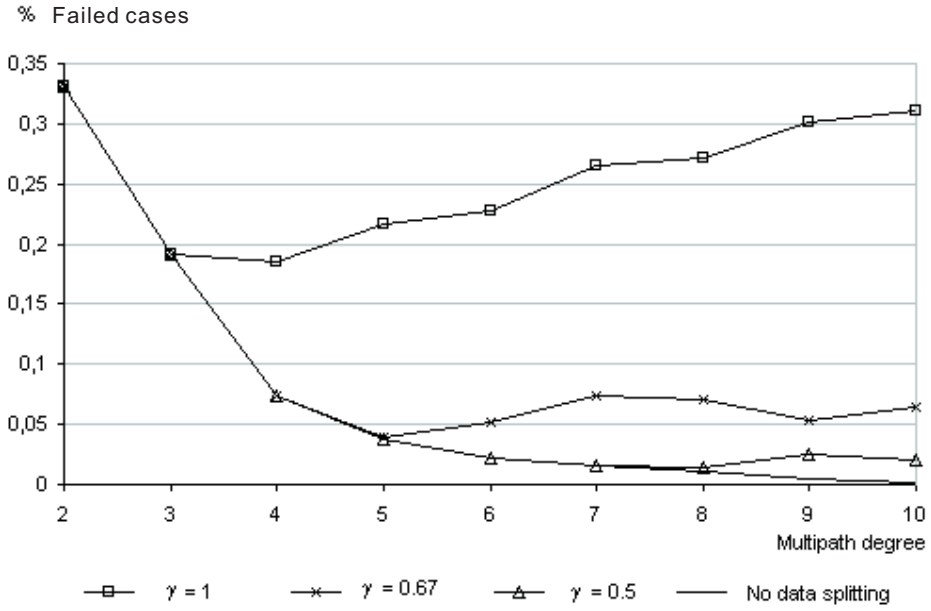


Figure 3.20: Failed transmissions percentage for different γ values

One of the interesting results of this work shows that, contrary to one's intuition, for a given maximum node failure probability, using a higher multipath degree than the optimum one actually increases the total probability of failures (see Figure 3.20 on page 69). This happens due to the way the algorithm computes the average number of successful paths E_n. So after the routing algorithm discovers a number of available paths, a optimal number of multipaths should be selected to split the data.

We have implemented this scheme and estimated the main characteristics. It greatly increases the reliability of packet delivery in wireless sensor networks, while keeping the total network traffic much lower than the traditional multipath routing. At the same time the latency of splitted multitpath routing is shorter than any retransmission scheme. There were several interesting observations that came up at the end of this study. The most important are:

- the speed of the nodes has little influence on the parameters of the algorithm. A way of diminishing the effects of mobility is usually increasing the transmission range of the nodes. This implies a higher energy consumption. By using multipath routing, this is not necessary. Intuitively this means that the same results can be achieved with a lower amount of energy.
- the failures can affect also the control messages. The solution to this problem

is modifying the route reply phase or even change it completely with the route reply phase of the DSR algorithm. Retransmitting the control messages incurs large delay in a unreliable network, so other solutions have to be explored.

Our data splitting scheme provides an abstraction of a better transmission medium from the receiver side and a retransmission scheme could further take care of the remaining error corrections. Our scheme could disseminate a large amount of bulk data to the destination with high reliability and low delay. In examples like streaming or code dissemination, our scheme transmits large amount of data which are composed of large packets to the remote party and with the ability to adjust the reliability while keeping the data traffic low. Although focused on WSNs, it can be incorporated into any routing scheme to improve reliable packet delivery in the face of a dynamic (wireless) environment where nodes move and connections break.

Chapter 4

Reliable Source Routing Protocol for WSN

In wireless sensor networks, the highly dynamic topology is caused by node mobility and dynamic changing wireless link quality. As the network size grows, the chances of routing failures in a multi-hop network greatly increases. The traditional layered networking approach in developing network protocols has several drawbacks in the resulting performance and efficiency of the system. This chapter presents a new approach for networking in wireless ad hoc and sensor networks, which addresses the reliability of dynamic wireless sensor networks in point-point routing scenarios by cross-layer interaction. This approach relies on a self-organizing medium access *(MAC) protocol, which uses an algorithm to decide the grade of participation of a sensor node to create a connected network based upon local information. On top of this connected network, a tightly integrated on-demand routing protocol is designed. It benefits from the local neighbor information of the MAC protocol and is able to efficiently maintain and recover routes based on interaction with the MAC protocol. In the route re-establishment, a directional and geographically restricted flooding scheme based on previous knowledge of the location of the destination node is devised. Both simulation and implementation results show that the cross-layer approach dramatically reduce energy consumption in a dynamic wireless environment. The work presented in this chapter is published in EWSN'04 [109]*[1] *and in IEEE Wireless Communication Magazine [39]*[2].

[1]Jian Wu, Paul Havinga, Stefan Dulman and Tim Nieberg. "*EYES Source Routing Protocol for Wireless Sensor networks*". In Proceedings of the First European Workshop on Wireless Sensor Networks, Berlin, Germany, 2004.

[2]Lodewijk van Hoesel, Tim Nieberg, Jian Wu, Paul Havinga. "*Prolonging the Lifetime of Wireless Sensor Networks by Cross-layer Interaction*". In the Special Issue on Wireless Sensor Networks, IEEE Wireless Communication Magazine, 11(6):78-86, 2004.

4.1 Introduction

In this chapter we presents a cross-layered approach for networking in wireless sensor networks. It addresses the reliability of dynamic wireless sensor networks in the point-point routing scenarios by cross-layer interaction. In WSN, a traditional layered networking approach in developing network protocols has several drawbacks in the resulting performance and efficiency of the system. Especially in the application area of high mobility, such as vehicle tracking and personal tracking, this problem is more significant. Our approach relies on a self-organizing *medium access* (MAC) protocol, which uses an algorithm to decide the grade of participation of a sensor node to create a connected network based upon local information. On top of this connected network, a tightly integrated EYES Source Routing (ESR) is designed. It benefits from the local neighbor information of the MAC protocol and is able to efficiently maintain and recover routes based on interaction with the MAC protocol.

Our lessons learned in developing network protocols for wireless sensor networks in the last couple of years show that using the traditional layered networking approach has several drawbacks in the resulting performance and efficiency of the system. Quite often, significant improvements are possible for the network protocols; yet they require a significant amount of information to be passed along the layers of the system. Although this approach allows in principle independency between the various protocols, it incurs a significant overhead in parameter transfer. Moreover, improvements performed in a specific layer can have impairments and even be counterproductive for other layers.

We will show in this chapter that optimization is more effective when taking into account the overall system, and with the use of all available knowledge. When this information has to be distributed to other sensor nodes, the effect is even larger. A solution in which such information is piggy backed to other messages can limit the extra message exchange. During the development of various protocols and services (like localization protocols), the lowest layers of our system (e.g., the MAC layer) were increasingly being used to pass information to these higher layers. The overall result of these developments has led to the cross-layered approach as described in this chapter. In this approach we first designed EYES Medium Access (EMAC) protocol, which is a self-organizing TDMA-based *medium access* (MAC) protocol. Then we design an algorithm to decide the active set of nodes, which create a connected network based upon local information. On top of this connected network, a tightly integrated on-demand routing protocol, EYES Source Routing (ESR), benefits from the local neighbor information of the MAC protocol and is able to efficiently maintain and recover routes based on interaction with the MAC protocol.

In Section 4.2 we give an overview of related works in this domain. The

SMAC protocol [113] and the DSR routing protocol [43], two protocols commonly used in WSN researches to compare MAC and routing protocols, are discussed in more detail. Section 4.3 discusses the design of the EYES Medium Access Control protocol (EMAC) and Lightweight Medium Access Control (LMAC) protocol, that is especially designed for WSNs and that allows us to exploit the benefits of the cross-layer approach discussed in this chapter. We pay special attention to the decision mechanism that sensor nodes use in Section 4.4 to either actively take part in the network or to save energy by using resources of the backbone nodes in the network. The designed EYES Source Routing protocol (ESR) is presented in Section 4.5. Section 4.6 presents the results of the simulation and Section 4.7 presents the implementation of our cross-layer protocols in a real test bed. Finally, Section 4.8 gives the conclusion of this chapter.

4.2 Related Work

As the cross-layer approach presented in this chapter involves medium access protocol and MAC-clustering, this section gives more introduction to these areas, which is an addition to the state of art in Chapter 2. We will first discuss the Sensor-MAC Protocol (SMAC), which we use later on in this chapter to compare results with our MAC protocol. Then we give an introduction of MAC-Clustering, which is related to our active set design. Finally, we discuss in more detail current routing protocols and identify the aspect we stress in this chapter.

4.2.1 Sensor-MAC Protocol for WSNs

Although the research field of WSNs is relatively new, some interesting studies of MAC protocols can be found in other literature. One of those protocols is the Sensor-MAC Protocol (SMAC), which we will use later on in this chapter to compare results with our cross-layer approach. This section gives a short introduction to this protocol.

The SMAC protocol [113], developed by Wei Ye, John Heidemann and Deborah Estrin, recognizes two phases in transceiver usage of network nodes: a listen period and a sleep period. In the sleep period, the nodes turn off their power consuming transceiver. After the sleep period, the nodes wake-up and listen to whether communication is addressed to them, or they initiate communication themselves. This implies that the sleep and listen periods should be (locally) synchronized between nodes. Since the protocol is *carrier sense multiple access with collision detection* (CSMA/cd) based in the listen period, synchronization does not have to be very strict and nodes can also use their sleep period for communication if needed.

Choosing and maintaining a sleep/listen schedule - Before each node begins sleeping and listening periodically, it has to choose a schedule and must exchange the schedule with its neighbors as follows:

1. A new node listens for a defined amount of time. When it does not receive a schedule from another node, it randomly chooses a time to enter the sleep phase and transmits this information (a relative time) in a so-called SYNC packet to its neighbors. The node now defines the schedule in the network and is called the synchronizer.

2. If a node received a SYNC message during the synchronization phase, it adjusts its schedule to the information in the SYNC message. The node follows the sleep/listen schedule in the network. A random time interval is waited before the follower transmits a SYNC message, in order to prevent collisions between SYNC packets when multiple nodes are triggered by the same SYNC message.

3. If a node has already chosen a schedule and becomes aware that one of its neighbors is following a different schedule, it keeps its own schedule and also wakes accordingly to the schedule of the other node. However, the node will not transmit its new listen and sleep pattern in SYNC messages. Instead, it just transmits its own chosen schedule to prevent a propagation of rescheduling in the entire network. Network nodes between regions with different schedules of listen and sleep periods have less sleep time compared to others and therefore their energy consumption is higher. All nodes maintain a table with the schedule of their neighbors.

Communication in the listen period: - To prevent collisions of short SYNC messages, which contain only an identification number of the sender and the next time nodes go to sleep, the SMAC protocol divides the listen period in two sections. The first part is reserved for SYNC messages and the other part is reserved for *request to send* (RTS) messages. The SMAC protocol is also capable of transmitting *omnicast* messages. These messages are not acknowledged by receiving parties.

4.2.2 MAC-Clustering

In order to decide which nodes have to remain active to ensure an operational and connected sensor network, we use ideas coming from clustering techniques. In the context of wireless sensor networks, clustering is mostly used to group the nodes for routing protocols. Clusters are usually controlled by a designated node called a *clusterhead*.

In [20], a distributed, randomized algorithm for dominating sets is presented. Several authors, e.g. [7, 30, 6] focus on clustering schemes where the clusterheads form an independent set in the wireless network. In order to obtain an overall connected structure, so called *gateways* are introduced that are used to create connections between the clusterheads. Basagni [7] presents a set of local protocols that create and maintain the set of clusterheads taking into account dynamic environments, e.g. due to node failures and mobility.

In this chapter, we aim for a combined protocol of the lower networking layers. The ideas from the clustering algorithms are applied directly to the MAC-layer in order to create a connected backbone of the network consisting of the active nodes. Many ideas and approaches coming from these clustering schemes that try to create a (maximal) independent set, or a (connected) dominating set, can be used in our approach [71]. Our mechanism provides nodes 1) the ability to be idle and in a low power mode for a long period of time, 2) the possibility to quickly use the communication infrastructure, and 3) that create an efficient and connected backbone.

4.2.3 Routing

This section gives more explanations and analysis of routing protocols for WSN, which is additional to the state of art in Chapter 2. There has been a very significant effort in research and development on routing mechanisms for wireless ad hoc networks. Related work on routing protocols for wireless sensor networks includes LEACH [35], Directed Diffusion [34], GRAdient Broadcast [111], etc.

LEACH is built on the assumption that all sensor nodes can directly reach the sink node by only one hop. Therefore, LEACH cannot scale up to networks with a large geographical size. Directed Diffusion is a data concentric routing scheme which relies on local interactions between sensor nodes to create efficient paths for data flow. Directed Diffusion does not scale well if the user is mobile, because the end-to-end 4-way handshake protocol between local nodes has to be repeated every time the sink moves. GRAdient Broadcast builds a cost field toward the sink and then reliably routing queries across a limited size mesh toward this sink. The performance of GRAdient Broadcast degrades significantly when the node, especially the sink moves, because network wide reflooding is required to adjust the cost field in the intermediate nodes.

In a WSN, data generated by one or more sources usually has to be routed through several intermediate nodes to reach the destination due to the limited range of each node's wireless transmissions. Moreover we assume that the topology of a WSN is dynamic due to node mobility, and due to the fact that nodes will be powered

off regularly in order to save energy. Routing protocols for ad hoc wireless networks, such as *dynamic source routing* (DSR, [43]) and *ad hoc on-demand distance vector* (AODV, [76]) are designed for dynamic networks in which the nodes come and leave periodically similar to sensor networks. In this chapter we will compare our routing protocol with DSR as a standard reference point.

In DSR, when the source has a data packet to send to another node, the destination, it requires a multi-hop route through the network. For this purpose, each node stores routes from previous paths in its cache. If there is no routing information to the destination in this cache, the route discovery process is initiated to create such a route. This is done by flooding the network with route request messages. Each node adds itself to a list to build up possible routes and rebroadcasts the message. When such a request reaches the destination node, it sends a route reply message back to the source. This is also done when an intermediate node has routing information to the destination stored in its cache. DSR also offers routines to detect broken paths and create new routes by sending route error messages upon detection of a broken link in the path during usage of that route. Nodes that receive a route error message cancel all routes that use the link from their cache. A new route discovery process is then started at the source node.

However, in DSR and many other routing protocols, route re-establishment, relying on flooding the whole network with requests, is required to recover the lost link. Such a measure would be significantly less efficient, as the average movement speed of the nodes and the diameter of the network increases. The new idea behind our routing algorithm is to recover the broken link in a fast and efficient manner such that the rate of energy consuming route re-establishment is suppressed to a low level. In our approach we are able to deal with topology changes efficiently because it uses information about its neighboring nodes that is already available and used in our MAC protocol.

4.3 Medium Access Protocol

In this section we introduce the MAC protocol designed for our cross-layer approach. Our routing protocol is able to deal with topology changes efficiently because it uses local information about its neighboring nodes that is already available and used in our MAC protocol. Moreover, the *Traffic Control* section of the MAC protocol provides an efficient way of message exchange for both routing and active node algorithm.

The costs of sensor nodes must be kept at a minimum. This does not only translate to scarce resources –like energy and memory– in the sensors, but also to

complexity of the hardware. Currently, multi channel transceivers are available on the market, but they will always be higher priced than single channel versions. During the design of the Medium Access Control protocol, we assumed such a single channel transceiver that has three operational states: transmit, receive and standby. Typically, transmitting consumes more power than receiving, and standby lies beneath the power consumption of receiving by a factor of 1,000 or more.

The next two parts describe the EYES medium access protocol (EMAC) and its successor, the Lightweight MAC (LMAC), which is used in our implementation.

4.3.1 EYES medium access protocol

The EYES Medium Access Control protocol (EMAC) [96] is based upon Time Division Multiple Access (TDMA). Time is divided into *time slots*, which nodes can use to transfer data without having to contend for the medium or having to deal with energy wasting collisions of transmissions. After the frame length, which consists of several time slots, the node again has a period of time reserved for it. But unlike the traditional TDMA, the time slots in our protocol are *not* divided among the networking nodes by a central manager. In the following text, we explain how the sensors can autonomously pick time slots with only local network knowledge.

We have three modes of operation in our MAC protocol: active, passive and dormant mode. When a node is in *active mode*, it will contribute to the routing by taking part in forwarding messages to a destination and accepting data from passive nodes. *Passive nodes* on the other hand do not actively participate in the routing process. They conserve energy by only keeping track of one active node, which can forward their data and informs them of network wide messages. The nodes in *dormant mode* put themselves in a low power state for an agreed amount of time or, for example, when their power source runs out of energy and has to be charged again using ambient energy, like light.

An active node performs three actions in its time slot. For a short fraction of the time it listens for incoming requests from passive nodes (in the so called *communication request* (CR) section). Next, it transmits a short control message (the *traffic control* (TC) section), which contains, besides a possible acknowledgement to the requests, other control and synchronization information, such as a slot schedule table. Nodes also listen to their neighboring TCs, and thereby obtain knowledge of their local neighborhood for their slot table. This knowledge is utilized for selecting appropriate time slots and used in the routing protocol. The remainder of the time slot, the *data section*, can be used for the actual transfer of data from higher network protocol layers.

A passive node neither controls nor claims a time slot. It chooses one active

node whose control messages it will listen to and files its requests, if any, to this active node. This allows for significant energy conservations in the passive nodes and the lifetime of the network is largely extended, especially if the role of active and passive nodes is changed over time.

Communication Request Section

Passive nodes do not control a time slot and therefore are not able to receive request from other passive nodes. The nodes can still transmit data by placing a request in the communication request section of an active node. Collisions of requests may occur, but the data rate in WSNs is low, so that we seldom expect a collision of requests. When a collision occurs, the active node is notified in its TC section. If the active node did not plan to use the data section for itself, it will allow the passive nodes to content for the medium in its data section.

The CR section consists of only a few bytes, namely: an identifier of the passive node and the request type. A special kind of request is the *node announcement*. This request is made by nodes that become active and want to notify other active nodes of their existence, so that they can participate in network activity.

Traffic Control Section

Every node in the network that is active, transmits a TC section in its time slot. This TC section is, so to say, the *heartbeat* of the network. Since these sections are always transmitted, new nodes in the network can use this section to synchronize to the time slot rate. The beginning of the TC section is precisely timed and the section contains timing information, i.e. the time slot sequence number and a measure of accuracy of the time of the node.

An active node that controls a time slot, identifies itself in the TC section with an ID number. Requests from other nodes in the CR section are acknowledged in the TC section. When a collision of requests is detected, the requesting nodes are notified by a flag in the TC section. An active node addresses another node in its local neighborhood by indicating a request in the TC section of its time slot.

For the spatial reuse of time slots, the nodes use an algorithm based on local information only. As explained above, the active nodes transmit a small table in the TC, which contains what time slots the node considers to be occupied by itself and its one-hop neighbor nodes. This information can be efficiently encoded by a number of bits equal to the number of time slots in a frame. Nodes can pick a time slot when the slot is considered to be free by all its neighbors. This method ensures that a time slot is only reused after at least three hops. Figure 4.1 on page 79 gives

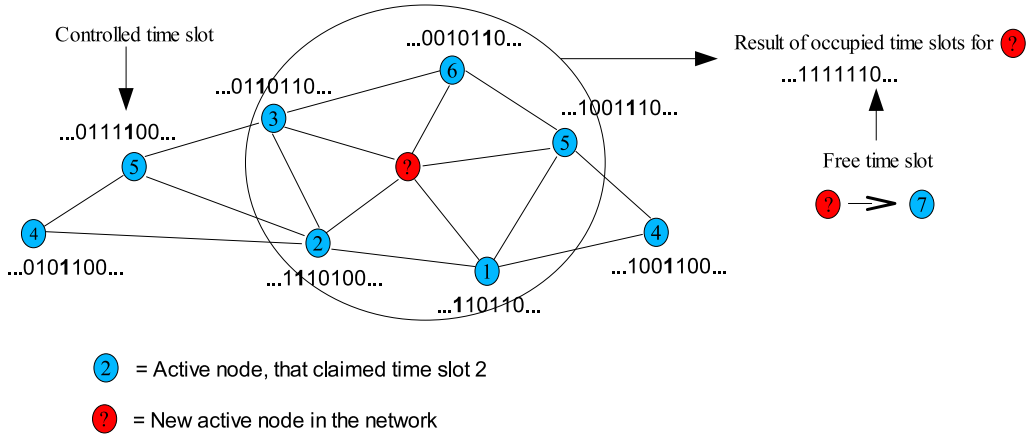


Figure 4.1: A new active node in the network can pick a time slot once it has discovered all its neighbor nodes

an example of how a new active node in the network can pick a time slot after it has discovered all its neighbors. Note that active nodes will only use their own time slot to transmit data.

From time to time, active nodes give up their time slot and re-execute the algorithm to pick a new time slot. This prevents collisions when active nodes travel through the network and meet another active node that has claimed the same time slot. When a sensor node is mobile, it should be given the preference to be a passive node in order to minimize degradation of network performance.

Routing protocols that allow messages to be routed over the ad hoc network typically require knowledge of the actual topology in order to efficiently route the packets over the network and deliver them to the destination. Storing routing tables in a static network might be an option, but when the topology is dynamic –like we assume– the resulting frequent updates are too energy consuming. By listening to TC sections of their neighbors, nodes have knowledge of the *local topology*. This assists routing and reduces the number of routing messages in the network. A special portion of the TC section is reserved to efficiently transmit the short omnicast messages that are generated by the routing protocol (see Section 4.5).

The TC section further contains a flag at the beginning of the section that indicates whether the section is different from the one transmitted in the previous frames by the active node. When the section has equal content, nodes that successfully decoded the TC of the time slot controller one frame ago, can turn off their energy consuming receiver immediately after the flag has been decoded. In many cases, this leads to large energy savings since the network can be idle for long times. The nodes can still use the very small part of the TC section they received

to synchronize with the network.

Data Section

The data section is reserved for the actual data transfer. The format and length – up to the CR in the following time slot – of this section is completely free for use by the higher layers in the networking algorithms. This section is long compared to the other sections. In our simulation we have chosen for a maximum data section length of 64 bytes because of memory constraints in the sensor nodes.

4.3.2 Lightweight Medium Access Protocol

In this section we present the Lightweight MAC (LMAC) protocol [97] which is a variant of the EMAC protocol and is implemented together with our routing protocol ESR on the sensor node prototype to evaluate the performance of the cross-layer approach. The intention of the protocol is to minimize the number of transceiver switches, to make the sleep interval for sensor nodes adaptive to the amount of data traffic and to limit the complexity of implementation. The MAC protocol is based on ideas from the EMAC protocol, which divides a time slot into three sections: Communication Request (CR), Traffic Control (TC) and the data section. In the CR section, other nodes can send a request to the node that is controlling the current time slot. However, as this section increases the amount of transceivers's use and will cause a collision, we omitted this section in LMAC.

Initially, a gateway exists in a network and has already occupied a time slot. When the nodes power on, they are all unsynchronized. They first try to discover the gateway in this network and then synchronize their clocks to it by listening to the control message of this gateway. Other nodes in multiple-hop distance from the gateway can also synchronize to nodes which have already synchronized to the gateway. Meanwhile, nodes choose an empty time slot as its slot to control.

Each time slot includes two sections: traffic control and data transmit. A node will always transmit two kinds of messages in two parts: control message and data unit. Each node in the neighborhood occupies different time slots, thus the communication among them is collision-free.

In the traffic control phase, nodes in their time slots can broadcast control messages to their local neighbors. The fixed-length control message consists of some MAC and routing information and is used for several purposes. Firstly, it carries the identity number of this current time slot controller and it indicates the distance of the node to the gateway in hops of the network. Next, the control message can transmit its slot usage known to maintain time synchronization between the

nodes, and records the possible collision information, so that nodes in collision can randomly choose their time slots again. It also addresses which nodes will received the messages from it in the data transmit section by the destination identity number and the size of received messages. Furthermore, it can pass the short multicast route message in the control message and broadcast all one-hop nodes, which will reduce the route control overhead and help to enhance the network efficiency. We assume that the clock drift is neglectable in a single frame, and the nodes may have clocks with low accuracy.

According to the control message received from the current time slot, other neighbor nodes know whether they should continue to keep its transceiver active to receive messages by listening to the destination ID. When a node is not addressed in this message and the message is not addressed as an omnicast message, the node will switch off its transceiver and enters a low-power standby state until the next time slot starts. Alternatively, when a node is addressed in this message, it will keep the transceiver on and listen to the message in the data section. If the time consumption of receiving the message doesn't cover the entire remainder of the time slot, both the transmitter and receiver(s) will turn off after the message transfer has completed.

After neighbors received all traffic control messages of a frame, they will have the knowledge of the local topology, i.e., which nodes are their one-hop neighbors, and record these nodes' Id in neighbor table including the sequence number of the time slot. This helps routing protocols to reduce the number of routing messages, so that the packets route efficiently over the network to the destination

4.4 Active and Inactive Nodes

In this section, we present an algorithm that is used to identify the nodes that actively participate in the routing tasks. The decision is taken locally according to information from the neighboring active nodes only. We present a local, distributed algorithm whose control information easily fits into the Traffic Control (TC) section of the two MAC-schemes presented in the previous section. This algorithm is part of the integral cross-layer approach taken for networking in wireless sensor networks.

The set of active nodes should form a *connected dominating subset* of the nodes in the sensor network [71]. A subset of nodes is called dominating if every node in the network is either in the set or can reach a node from this set by direct transmission. Thus the active nodes who know about the inactive nodes in their neighborhood, may hold data until the recipient can be woken up and notified.

Since inactive nodes do not actively participate in the routing process of the

Table 4.1: Roles of a node ID

Description	Encoding
Anchor node	AID = ID
Bridge	AID = (Anchor1 XOR Anchor2)
Undecided Active	$AID = 0$
Nonmember	AID = Lowest-ID Anchor

network, the set of active nodes is required to form a connected set. This way, each node of the network can eventually be reached by an ad hoc routing process.

This set of active nodes is from now on referred to as the *connected active set* of nodes. Nodes that need to be active to ensure the above properties are contained in this set will be referred to as *active* for the remainder of this section. Nodes that are not in this set are *passive* nodes. Note that passive nodes may use a time slot and participate in the network.

4.4.1 Roles and their encoding

In order to decide which nodes are active and passive, several roles are given to the nodes that are participating in the network. Nodes that own a time slot periodically transmit a TC section, notifying all surrounding nodes about their neighbors and their *AID*. This *AID* indicates what role the node is performing with respect to the connected active set.

These roles are given in Table 4.1 on page 82, together with their encoding in the *AID* field of the TC section. The *anchor* nodes are locally assigned to cover the network so that no two anchor nodes are direct neighbors. If an anchor node can reach (via other active nodes) all anchor nodes that are at most three hops away, the entire set of active nodes is connected. To achieve this, *bridge* nodes are introduced. There are two types of bridging nodes. A node that receives the TC sections of two or more anchor nodes is called a *direct bridge*. If two intermediate nodes are needed, these two nodes form a *distributed bridge*.

For the *AID* field, the first bit when using node IDs is always set to 0. This is done to identify bridges, which then have a leading 1 in the *AID* field. Also, the value given is there to avoid mistaking a possibly nonexisting node ID. Nodes that are not part of the connected active set (passive nodes), but participate in the network by owning a TC section, are identified by having an *AID* corresponding to the neighboring anchor node with the lowest ID. This encoding also helps in identifying distributed bridges.

A special role is given by *undecided active*, which is mainly used when a node enters the network, e.g., by waking up, and has not found a neighboring anchor.

Generally speaking, the anchor nodes form the main part of the connected active set and are spread out and maintained over the sensor network. The bridging nodes are formed to connect the adjacent anchor nodes.

4.4.2 Local decision algorithm

Each node that enters the network, e.g., by waking up or being deployed, has to decide whether it is needed as part of the connected active set. This is achieved by the following algorithm. Additionally, this decision process is performed when a change in the local topology given by the active nodes occurs. This is witnessed by a change in a frame.

A schematic overview on the decision algorithm that is run in each node upon deciding is presented in Figure 4.4.2 on page 84. Next, we present the individual steps and decisions in more detail.

- **Neighboring anchor** – If there are neighboring anchors, the node cannot become an anchor itself. However, if there is no anchor identified, the lowest ID criterion is used to elect an anchor. For this, a node checks whether it is the undecided active node with the lowest ID in its neighborhood and becomes anchor node if this is the case. Otherwise, it waits for undecided nodes with lower IDs to decide first. This follows the idea of *lowest-ID* clustering [30].

- **Bridging decision** – If there are two or more anchor nodes in the neighborhood, a node checks whether there is already a direct bridge in the neighborhood connecting pairs of anchor nodes for which the XOR is also locally computed.

- **Distributed bridging decision** – For a node to become a distributed bridge, one of the anchor nodes is not in its neighborhood. This can be determined if there is a neighboring nonmember node whose *AID* is not in the neighborhood. In that case, these two nodes can form a distributed bridge. Each node locally stores the ID of the node it forms a distributed bridge with.

- **Become passive** – A node that determines that it is not needed in the connected active set, does not drop out of the process immediately. For the next frame, it transmits its neighboring anchor with the lowest ID for distributed bridging detection. If after that no change in the neighborhood is detected, it can become inactive.

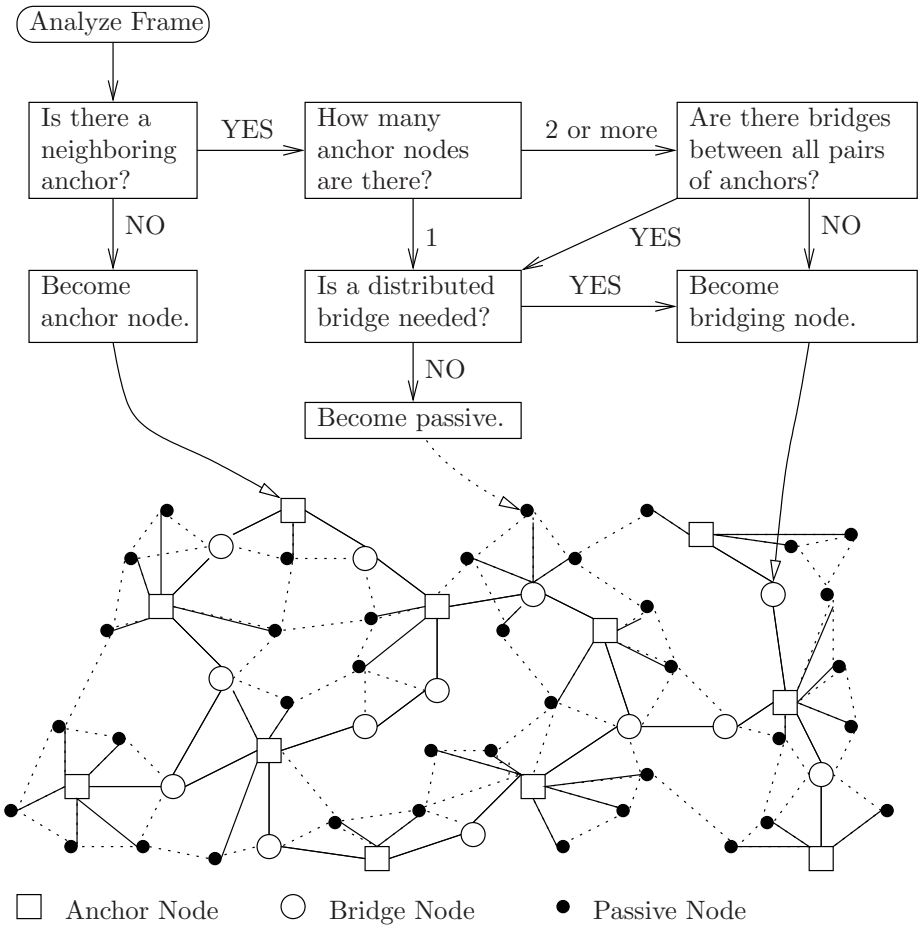


Figure 4.2: Internal decision algorithm of node to become active.

Note that if there are undecided nodes, the undecided node with the lowest ID in the neighborhood is always able to decide on its role as something other than undecided. The undecided role is thus only a temporary one.

Obviously, a node that participates in the network as part of the connected active set consumes more energy than passive nodes. Therefore, the principle of role rotation is supported in our scheme. An active node can drop its status and become inactive. Surrounding nodes will detect this and adapt by creating a new anchor or bridge if needed for connectivity.

4.4.3 Discussion

The structure created by anchor nodes forms a maximal independent set, which is also a dominating set of the network, and bridging nodes are introduced to ensure connectivity.

Especially in a dense network, many nodes are capable of performing the connecting duties of bridges. In our approach, only a few bridges actually have to remain active, as other nodes in the area realize their redundancy by the *AID* field. Thus, overall we obtain a connected dominating set given by the active nodes that uses only few nodes.

Nodes that need not be active, but have to participate in the network due to other reasons like actively reporting of sensor data are naturally supported.

4.5 EYES Source Routing Protocol

In the WSN, data generated by one or more sources usually has to be routed through several intermediate nodes to reach its destination due to the limited range of each node's wireless transmissions [10]. However, the topology of WSN is highly dynamic due to frequent node mobility. As the network diameter grows, the chance of intermediate nodes failure to forward the incoming messages greatly increase.

For the conventional routing protocols for ad hoc networks, such as Dynamic Source Routing protocol (DSR) [43] and Ad Hoc On-Demand Distance Vector protocol (AODV) [76], route re-establishment by flooding the whole network with requests is required to recover the lost link. Such a measure becomes significantly less efficient while the average movement speed of the nodes and the diameter of the network increase, which is shown in the simulation results of the next section. The new idea behind our routing algorithm is to recover the broken link in a fast and efficient manner such that the rate of energy consuming route re-establishment is suppressed

to the minimal level. The design of our algorithm is based on DSR and AODV and we keep the algorithm simple and lightweight, considering the resource limitation of tiny wireless sensors.

The EYES Source Routing algorithm[3] is an on-demand algorithm, which enables dynamic, self-starting, multihop routes to be established when a source sensor node wishes to send a data packet. We keep all the routing messages in ESR as small and fixed length packets. This is especially important in the *Route Setup* phase, where the source node initiates a request flooding and establishes a shortest path to the destination node. Because our route request can fit into the TC section of EMAC/LMAC protocol, no data section broadcast is needed in the route request propagation. In the *Route Maintenance* phase, ESR benefits from the local neighbor information of the EMAC/LMAC protocol. The algorithm resorts to local and fast recovery mechanisms to counter the unreliability of the established link and maintains the route between source and destination. When Route Maintenance fails, the route has to be restored. An effective request flooding was proposed in the Route *Re-establishment* phase. In the following sections, we give detailed introductions to the three phases of ESR.

4.5.1 Route Setup

Initially, when a node wants to send a data packet to another node, prior knowledge of the destination location is not available. In this stage, the source has to flood the whole network with route requests in order to notify the destination that it has a packet for it. All the nodes in the network are invoked to rebroadcast one route request, which is energy consuming. However in the *Route Re-establishment* phase, a new scheme was devised to maximally reduce the propagation of the request to a limited area, which will be explained in the later section. In the setup phase, the length of the request is small and constant to minimize the energy required in the flooding. The route request message consists of the following fields:

- *sourID*: the ID of the Source node
- *destID*: the ID of the Destination node
- *seq*: the sequence number of the request
- *lastID*: the ID of the last hop node

[3]This work is performed as a part of the European EYES project (IST-2001-34734) on self-organizing and collaborative energy-efficient sensor networks. Therefor we named the new algorithm EYES Source Routing (ESR).

- *HTL*: hops To Live

Each node in the network has a unique ID and each message source maintains a sequence number for the routing request it sends to a specific destination. So the combination of *sourID*, *destID* and *seq* uniquely identifies a route request in the network, from which a node can immediately determine that a request is being received for the first time or being a passive acknowledgement or just a duplicated request. The Hops To Live field can be used to limit the initial flooding to a maximal allowed diameter. A node discards any duplicated request, which is identified by the first three fields. If the request is new, the node creates an entry for this source-destination pair and stores the last hop ID of this request as the best neighbor to the source node. Then it replaces *lastID* with its own ID and rebroadcast the route request with *HTL*-1. After the initial flooding, each node in the network, or at least in the maximal allowed diameter, has the knowledge of its own best neighbor to the source node. The destination node replies to the first received request and discard the duplicated ones. It sends back a route reply which only confirms the nodes on the best route to the source. The other nodes in the network will delete the route entry of this source-destination pair after no reply is received within a specific waiting time and release the values stored in the senor node. The route reply message consists of the following fields:

- *sourID*: the ID of the Source node
- *destID*: the ID of the Destination node
- *seq*: the sequence number of the request
- *lastID*: the ID of the last hop node
- *nextID*: the ID of the next hop node

The destination node learns its best neighbor to the source from the *lastID* of the route request and generates the route reply, in which it puts its own ID as *lastID* and its best neighbor to the source as *nextID*. So a node only forwards a reply message if it is in the *nextID* field of the reply. In this way, the reply only travels back through the best route and each node along this route knows both it best neighbors to the source and destination after the route setup phase. Any data packet between source and destination can then be sent without the complete route inside and each intermediate node makes route decisions according to its own best neighbor pair, which reduces the routing overhead during data packet transmission.

4.5.2 Route Maintenance

The nodes in WSN can be highly mobile and the movement is unpredictable. Moreover the unreliability of sensor nodes can make them fail from time to time. In such a dynamic topology network, multihop links have a high frequency and probability of breaking down [80]. Constantly re-establishing the lost link by re-flooding the network with route requests can be quite costly in the respect of energy when the average speed of the node increases. In ESR, a novel approach based on HTL is introduced in order to recover the lost link in a local and fast manner, so that the frequency of network wide route re-establishment is significantly reduced. Two kinds of scenarios exist in the route maintenance stage and they are dealt with in the following section.

A **Route Re-catch** message will be sent when the node notices that its next hop best neighbor floats away from its transmission range.

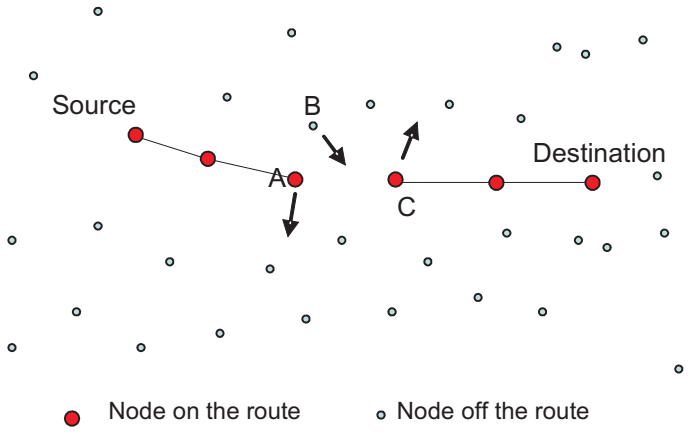


Figure 4.3: Links break when a node moves away

As shown in Figure 4.3 on page 88, when node A and C move in the relatively opposite direction, the radio link will break at a certain moment. Node A will notice that its best neighbor to the destination is no longer available. This could possibly be the result of no passive acknowledgement from the next hop or no response from periodical route update messages. The most effective approach to solve the route maintenance problem is in combination with MAC protocol, such as our EMACS and LMAC protocol, which are aware of all the neighboring nodes. To restore its next hop best neighbor, Node A sends the *Route Re-catch* message, which consists of the following fields:

- *sourID*: the ID of the Source node

- *destID*: the ID of the Destination node
- *catcher*: the node which sends the re-catch message
- *lastID*: the ID of the last hop node
- *HTL*: hops To Live.

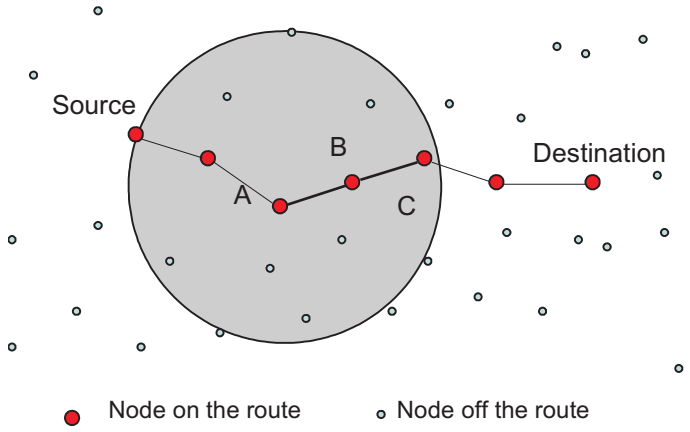


Figure 4.4: Locally restricted Route Re-catch process

To restore its next hop best neighbor, the node upstream sends the Route Re-catch messages locally, whose HTL is set to a very small value, depending on the average speed of the nodes and the density of the network. In our example, Node A adds its own ID to the *catcher* and *lastID* field. Then it broadcasts the *Re-catch* message with the HTL set to a very small value, which depends on the average speed of the nodes and the density of the network. In the example, the HTL sets to one. Any node receiving a Route Re-catch message, checks whether it is on-route from the *sourID* to *destID*. If not, it records the *lastID* as its best neighbor to the source and then forward *Route Re-catch* messages if the *HTL*>0. In the forwarded message, it sets its own ID as *lastID* and decreases the *HTL* by one; if yes, the node will sent a *Re-Catch Reply* back to the catcher with the following fields:

- *sourID*: the ID of the Source node
- *destID*: the ID of the Destination node
- *catcher*: the node which sends the re-catch message
- *lastID*: the ID of the last hop node

- *nextID*: the ID of the next hop node

So that this reply can travel back along the best restored route the same way as the *Route Reply* message explained in Section 4.5.1. After the "catcher" received the reply message, the broken link is restored successfully.

The *Route Re-catch* process can be viewed as an intermediate node initiated Route Setup. The difference is that firstly, the "source" in the Route Re-catch process is the node which sends the re-catch messages and the "destination" could be any on-route node downstream. Additional measures could be implemented to prevent an upstream on-route node sending replies. Secondly, the re-catch messages are limited locally to a very small diameter set by the *HTL*, as shown in Figure 4.4 on page 89. As the propagation speed of routing messages are much faster than the movement speed of the node, a properly selected *HTL* field has a high probability of catching the lost link. Thus, the routing algorithm is able to locally restore the broken link in a fast and efficient way, which greatly reduces the frequency of network wide flooding.

A **Route Cut** message will be sent when the node notices that its second order upstream neighbor comes into its transmission range.

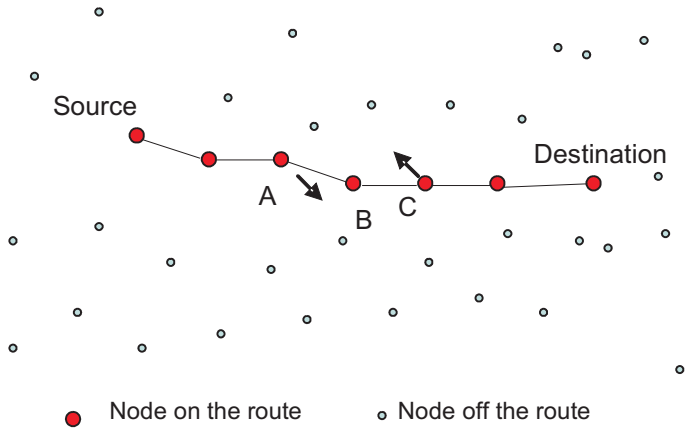


Figure 4.5: Link overlaps when node moves close

As shown in Figure 4.5 on page 90, when node A and C move toward each other, they will get into direct radio range. Node C will notice that Node B is no longer its best neighbor to the source. This could possibly be the result of overhearing data transfer from Node A. To change its next hop best neighbor, Node C sends the Route Cut messages, which consists of following fields:

- *sourID*: the ID of the Source node

- *destID*: the ID of the Destination node
- *sender*: the node which sends Route Cut message
- *newN*: the ID of the new best neighbor
- *oldN*: the ID of the old best neighbor

When Node A receives the Route Cut message, it changes its best destination neighbor from B to C and forwards the following data packet to node C. Node B changes its state to off-route when it receives the cut message. Although only one simple message, it effectively shortens the redundant link in the route maintenance.

4.5.3 Route Reestablishment

Route Re-establishment is necessary when local route re-catch fails and Route Maintenance is not able to recover the broken link. Although we still have to find the location of the destination node, the situation is different from Route Setup stage, in which no information could be used to help locate the destination node. The network wide flooding of requests is inefficient and energy consuming. However, in the Route re-establishment phase, temporary routes stored in the on-route nodes are valuable and need to be explored. In our algorithm, a directional and geographically limited flooding is proposed to reduce the energy required in the *Route Re-establishment*.

When the source node receives the error message sent back by the intermediate node that has not been able to recover the broken link, it tries to re-establish the route to the destination. One important observation is that although nodes in the network have mobility, its maximum speed is limited and normally not very fast in the WSN environment. As a result, the on-route nodes are somehow in the vicinity of the would-be shortest route to the destination, which will be re-established by the source node, as shown in Figure 4.6 on page 92. If the source could direct the request flooding along the old on-route nodes, then it could reach the destination with fewer number of messages.

In the algorithm, the Hops To Live field is used to control the direction and scale of the request flooding. When the source sends a Route Request in the re-establishment, it sets the HTL to a small value, which is enough to reach the next old on-route node. In the example, it sets to three. When a normal off-route node receives the request, it follows the procedure described in Section II.A to forward the request and decreases the HTL by one. Since the HTL is small, the flooding will stop in a matter of few hops. However, if the old on-route node receives the request, it acts as a HTL repeater by enlarging the HTL to its source value and then forwards

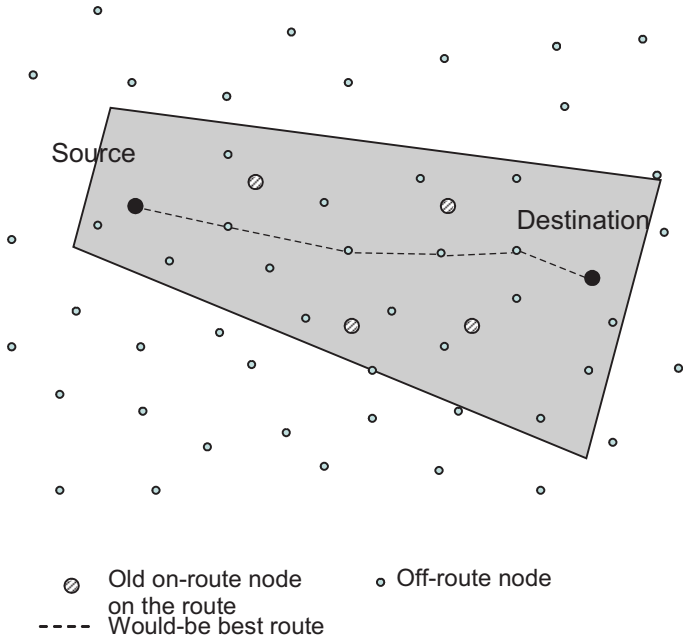


Figure 4.6: Directional flooding in the geographically limited area

the request. When this process occurs, the request flood will be directed to reach its destination. In Figure 4.6 on page 92, it shows that whenever the request flood encounters an old on-route node, it is enlarged along the direction of the destination. The overall effect is a destination aware and directional request flood, which soon dies out without the repeater effect of the old on-route nodes. It can be seen that the directional flooding is efficient compared with network wide flooding and the effect is more advantageous as the network diameter grows.

4.6 Simulation and results

4.6.1 EYES Source Routing Protocol

Simulation setup

The EYES Source Routing protocol was simulated on the OMNeT++ simulator [1]. In order to compare ESR with previous related work, we chose two routing protocols. The first one is Dynamic Source Routing (DSR) protocol[43], which is the ad facto standard for ad-hoc routing protocol and the other one is Ad hoc

On-demand Distance Vector (AODV), which has the basic on-demand mechanism of route Discovery and Route maintenance from DSR, plus the use of hop-by-hop routing, sequence number and periodic beacon. The simulation setup and parameters are identical for all protocols.

The nodes are initially scattered randomly within a square flat space of $600m \times 600m$. The transmission range of the sensor nodes are set to 150 m. Each run of the simulator accepts as input a scenario file that describes the exact starting location of each node and the random seed for each change in motion or packet origination to occur. A total of 20 different input scenario files with varying network size and movement patterns were generated and then all three of the routing protocols were run against each of these scenario files. The number of nodes are 25 nodes for a sparse network and 50 nodes for a dense network. During the 900-second simulation, all the nodes are free to move anywhere within this area. Each node chooses a speed from a uniform distribution between 0.1 and 10m/s and then moves to a random spot with the square space. When it reaches that spot, the node takes a rest for a specific Waiting Time (WT) before it moves again to a new spot with a new speed. In this way, a mix of moving and static nodes is achieved. It means from the whole network point of view, that at any given moment part of the network is moving and the rest of the nodes are in a waiting state. The length of the different waiting time decides how dynamic the network is. Ten source nodes randomly select their destination and sends data with a constant bit rate (CBR) of 4 packets per second, with packet size of 256 bytes.

Results and discussion

In order to compare the performance of the three routing protocols we evaluate them with respect to the following metrics:

- *Route acquisition time*: The time it takes a source node to discover a route to a destination node.
- *End-to-end delay (mean overall packet latency)*: The average times it takes for a packet to travel between the source node and the destination node.
- *Routing overhead*: The total number of routing packets transmitted during the simulation in bytes.
- *Throughput (packet delivery ratio)*: The ratio between the number of packets correctly received by the corresponding destination nodes and the number of packets sent out by the source nodes.

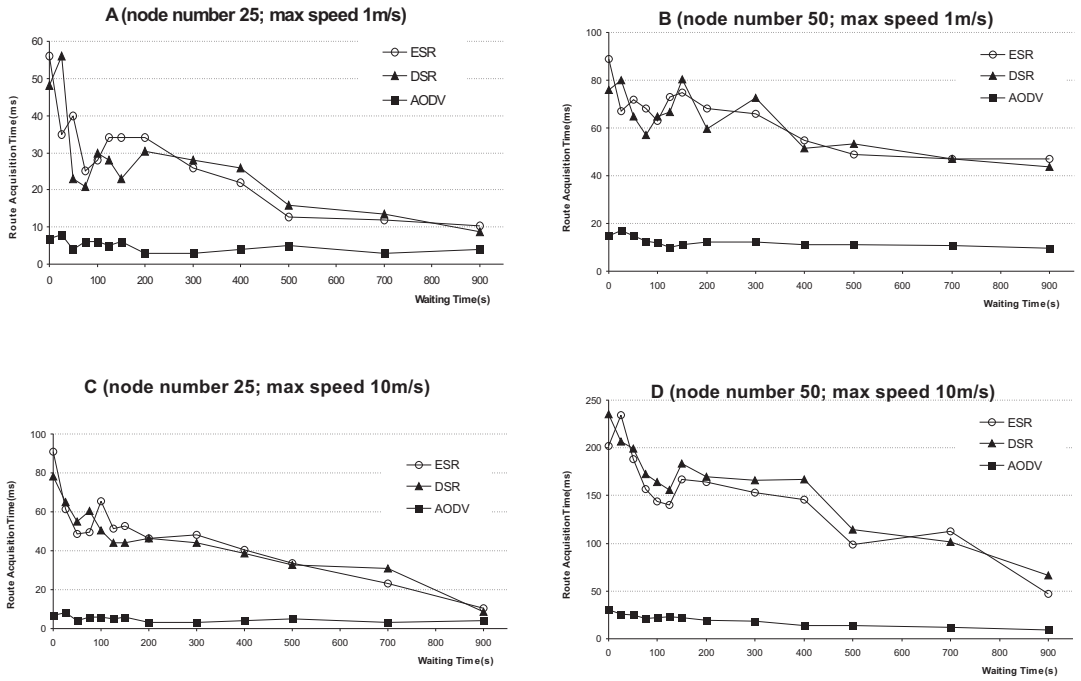


Figure 4.7: Route acquisition time for different mobility patterns (modelled by waiting time and speed)

The performance metrics for each algorithm depend heavily on the simulation scenario. But from the simulation results some trends could identify the characteristic of the routing protocols.

Figure 4.7A-D on page 94 gives a comparison of route acquisition time under different mobility patterns by the three protocols. As all the protocols are reactive routing protocols, they show an increase in route acquisition time for increased network speed, as many packets will be in an interface queue, while routing protocols try to find a valid route to destination. DSR and ESR are purely on-demand routing protocols. They rely on network wide flooding to find the destination node when there is data for it. AODV, as a combination of DSDV+DSR, tries to be aware of its link status with neighbors, thus route acquisition time is relatively low.

Figure 4.8A-D on page 95 shows the graph of average end-to-end delay under different mobility patterns by the three protocols. The increase of movement speed and decrease of waiting time, both induce a changing frequently topology and therefore increase the probability of broken links. Broken links may cause additional route recovery processes and route discovery processes. For this reason, the average

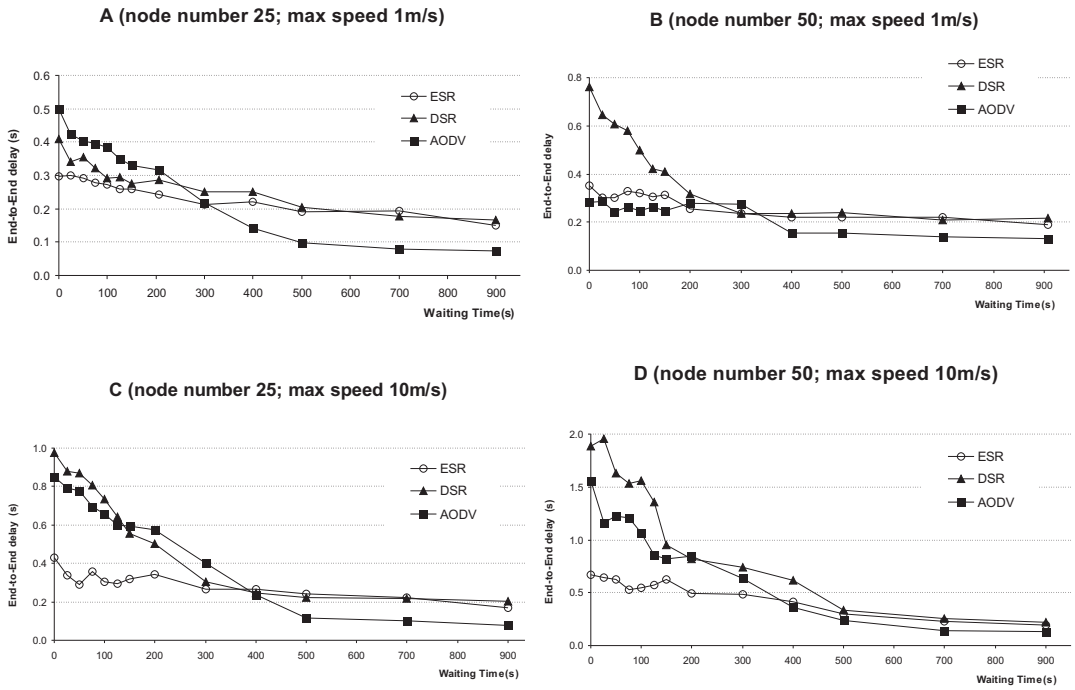


Figure 4.8: End-to-End delay for different mobility patterns (modelled by waiting time and speed)

end-to-end delay of packets increases as the node speed increases. In relatively static traffic (low speed, large waiting), AODV outperforms DSR and ESR in end-to-end delay as it has a small route acquisition time. In DSR, route recovery is fast, therefore it shows a better delay performance than AODV at small waiting time (high mobility) (Figure 4.7A and Figure 4.7B on page 94). But in high collision traffic (high speed, small waiting time) DSR control messages get lost, thus eliminating its advantage of quickly establishing new route. However as ESR has a faster local recovery mechanism, it restores the broken link locally and network wide route recovery is not needed. So it performs well under both cases. Under such situation DSR and AODV have a relatively high end-to-end delay, compared to ESR, but the delay decreases with increased waiting time (Figure 4.7C and Figure 4.7D on page 94).

Figure 4.9A-D on page 96 shows the graph of routing control overhead under different mobility patterns by the three protocols. The control overhead determines the scalability of the protocol, its performance in low-bandwidth environments and its efficiency in terms of energy consumption. From these graphs we see that the route control packet overhead increases with the increasing number of nodes, mobility rate and movement speeds. ESR showed the best performance, having a

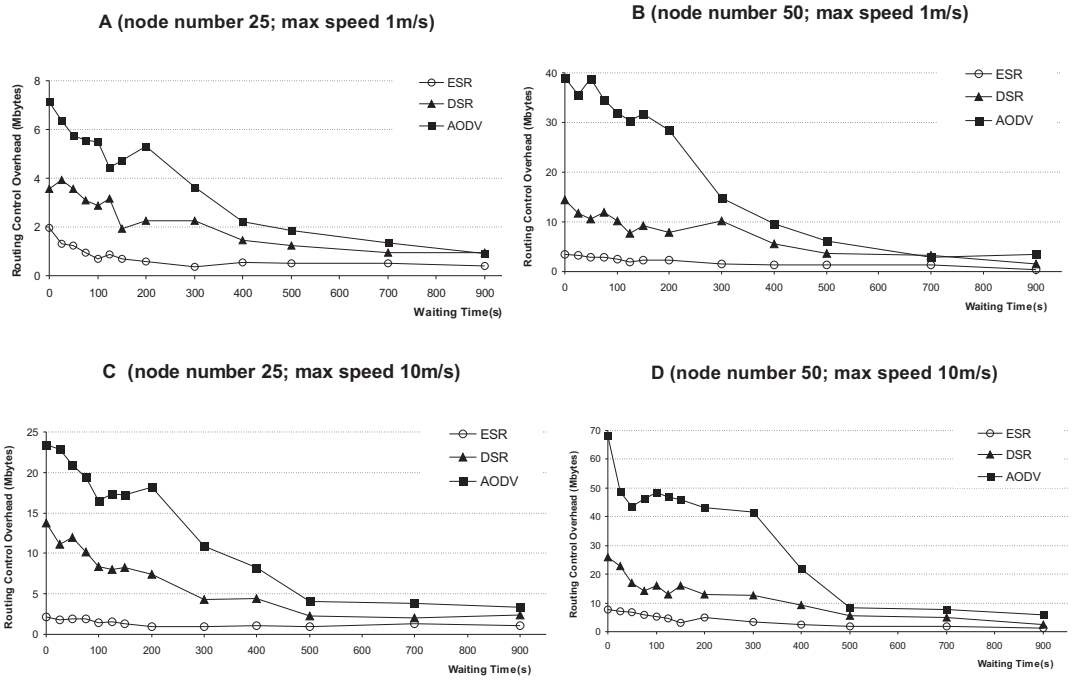


Figure 4.9: Routing control overhead for different mobility patterns (modelled by waiting time and speed)

relatively low routing overload for all cases. Especially with high speed, short waiting time and large networks, it has only about 40% of the overhead of DSR protocol and 20% of the AODV protocol as shown in Figure 4.9 D. The overhead for DSR is fairly constant despite movement rate or offered load.

Figure 4.10A-D on page 97 gives the comparison of data throughput under different mobility patterns by the three protocols. Network layer throughput describes the loss rate as seen by the upper layer. It shows the completeness and correctness of the routing protocol. These graphs show that throughput decreases for increase in network speed and mobility rate. This is because, at higher speeds and in more dynamic networks, more frequent link breakage may occur and therefore the packet loss rate increases. DSR and ESR routing protocols show better performance than AODV. ESR uses source routing and local recovery, that can reduce the total number of route control packets, resulting in less network congestion. Moreover its directional route rediscovery, which uses location information, can limit the broadcasted zone of route control packets. These characteristics can lead to better performance regarding to packet delivery rate.

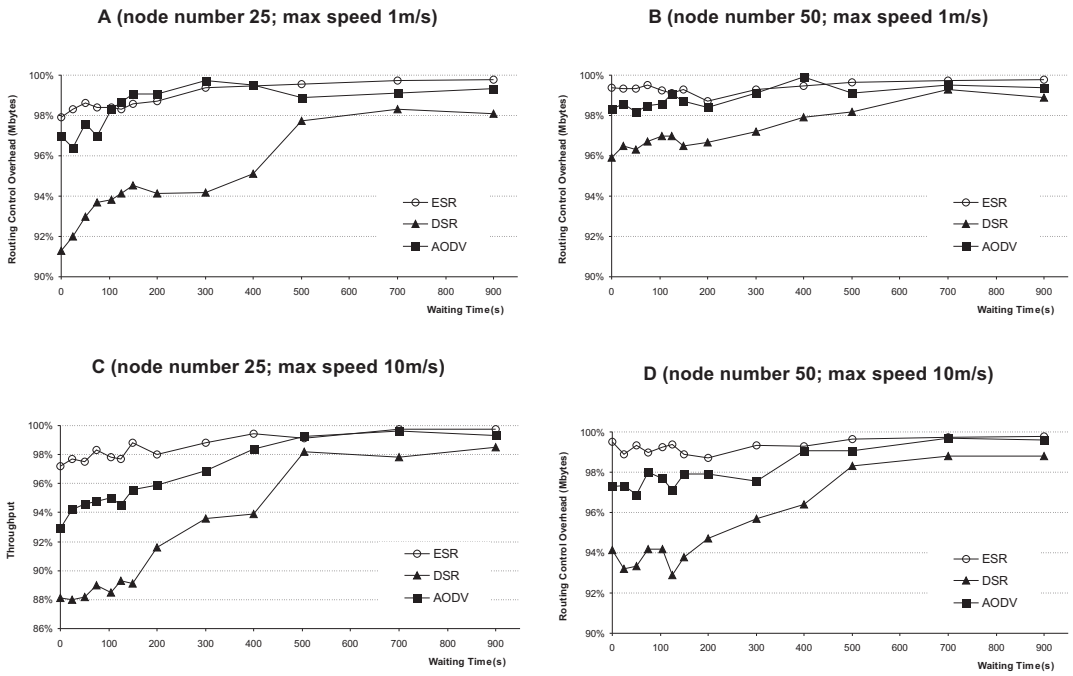


Figure 4.10: Throughput for different mobility patterns (modelled by waiting time and speed)

In Table 4.2 on page 98, we summarize the simulation results of the three protocols under different network speeds and densities. It shows that ESR achieves improved throughout performance over conventional routing algorithms in a dynamic network, while reducing power consumption on routing control overhead. We also notice that the performance gains of ESR become more significant when the network density increases.

4.6.2 Cross-layer approach

For the simulation of our combined cross-layer optimized networking protocols, the OMNeT++ discrete event simulator, together with a framework for a mobile, wireless network, is used. We compared the protocols presented in the previous sections with DSR and SMAC. The same network setup is used for comparing the two implementations of medium access and routing protocol.

In the simulator, a physical layer with energy model is implemented to record the sending and receiving energy consumption of the transceiver. Additionally, switching between sending and receiving takes time and consumes energy, which is

Table 4.2: The performance comparison between the routing protocols (good: +; medium: o; bad: -)

Parameters	Route acquisition time		End-to-End delay		Routing control overhead		Throughput	
Network dynamic	High speed	Low speed	High speed	Low speed	High speed	Low speed	High speed	Low speed
ESR	-	o	+	+	+	+	+	+
DSR	o	o	o	+	o	+	o	+
AODV	+	+	-	+	-	o	-	o
Network density	High dens.	Low dens.	High dens.	Low dens.	High dens.	Low dens.	High dens.	Low dens.
ESR	-	o	o	+	o	+	+	+
DSR	-	o	-	+	o	o	+	o
AODV	o	+	+	+	-	o	+	o

also considered in the simulation. The respective data for the transceiver is taken from an RFM TR 1001, which is also used in our prototype sensor nodes (see Table 4.3 on page 99). Although our prototype design can adjust its transmission range, we only consider the sending strength to be fixed to a high level, which yields an approximate coverage radius of 150 m.

The density of the resulting network is controlled by scaling the model, reducing the area for node placement yields a denser network, i.e. more neighbors within transmission range. The nodes move in this area according to the random way-point model (RWP) with random speed and waiting times. A mix of mobile and static nodes is achieved by adjusting the waiting time of the RWP model. A node that has reached its destination does not immediately pick a new way-point, but waits for a given period of time before moving again.

We use the **Network Lifetime** as the metric to evaluate the performance of our cross-layer optimized protocols. In wireless sensor networks, the metric of actual interest is not the transmission energy of individual packets, but the total operational lifetime of the network. Network Lifetime measures the amount of time before a certain percentage of sensor nodes run out of battery power.

Table 4.3: Transceiver data (RFM TR 1001)

Parameter	Value
Energy consumption transmit	21 mW
Energy consumption receive	14.4 mW
Energy consumption standby	15 μW
Switch time transmit/receive	518 μs
Switch time receive/transmit	12 μs
Switch time standby/receive	518 μs
Switch time standby/transmit	16 μs

Results

For the active connected set, we performed additional simulations to show only the results of the decision algorithm. In these simulations, we did not look at the routing mechanism and the node mobility and energy models were turned off. In order to find the relationship between the network density and the percentage of active nodes in the network, we have 2,500 nodes distributed randomly in a square area with side length of 1,000 m. The transmission range was varied to obtain different network densities. The active nodes, together with the percentage of anchor and bridge nodes, are given in Figure 4.11 on page 100. It shows that with low network density (average degree of neighbors) 43% of the nodes need to be active and with high network density the number of active node decreases to as low as 6% of all nodes. This means that in a dense network, only 6% of the nodes need to be active in routing.

In network lifetime simulation, 45 sensor nodes are randomly placed in a rectangle area of 800 m x 800 m. Five of them are chosen to be source nodes, which actually produce sensing data. The length of a data packet is 5 bytes and the data rate is varied in different simulation runs.

One (active) node is designated to be the data sink, which receives the data from these source nodes. The nodes move in the area according to the random way-point mode with random speed (2-10m/s) and waiting times (10-30s). A node that has reached its destination does not immediately pick a new way-point, but waits for a given period of time before moving again. In this way, a mix of moving and static nodes is achieved. Both the data sources and sink have an infinite energy budget, so they will not affect the network lifetime. During the simulation, when 30% of the normal sensor nodes are depleted of battery power, the whole network is considered to be down.

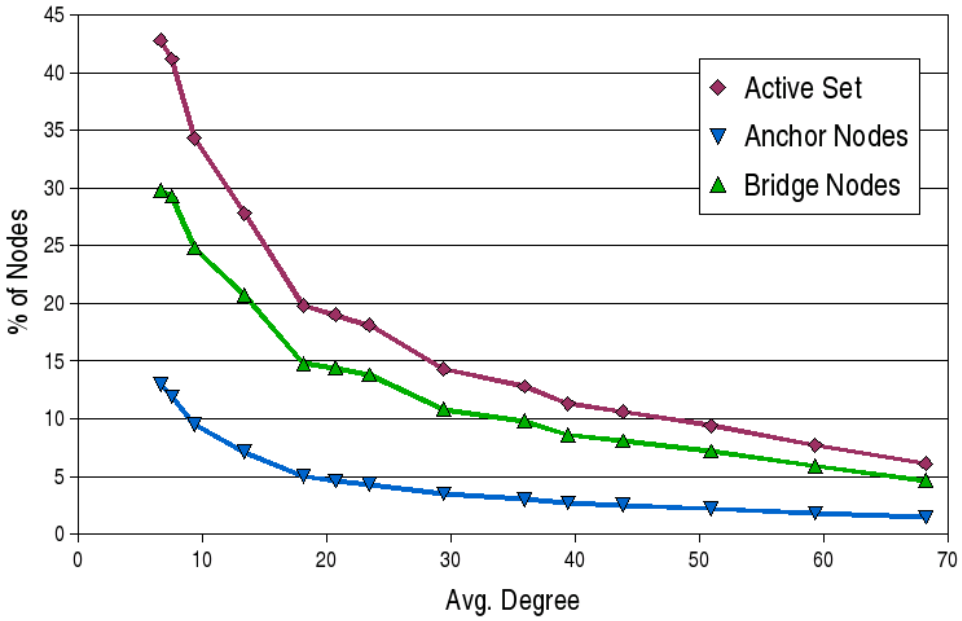


Figure 4.11: Percentage of active nodes in a network of 2,500 nodes

Figure 4.12 on page 101 shows the network lifetime of our approach and the reference DSR and SMAC, under different network loads from 0.5 to 50 packets/s. Note that the graph is normalized to SMAC and DSR in the static scenario.

It is shown that our cross-layer scheme prolongs the lifetime of the network significantly in the mobile scenario. A lifetime of at least 3 times the lifetime of DSR and SMAC could be reached. Although our protocols were designed to be efficient in dynamic networks, we also compared the protocol performance for static networks. In that scenario our scheme extends the lifetime 25% to 50%.

For comparison, DSR on top of EMAC is also presented in the graph to show the effect of routing on the lifetime. It shows that in static case, ESR has almost the same network life time as DSR. However, in a mobile network ESR has double the network life time compared with DSR.

Discussion

It is interesting to see that the lifetime, in the case of SMAC and DSR is not very dependent on the network load. This can be explained by the fact that the nodes always keep their receiver on in the time interval they are awake. The additional energy, which is necessary to exchange messages at relatively large intervals in our simulations, is negligible compared to the energy used in the listen period. In fact,

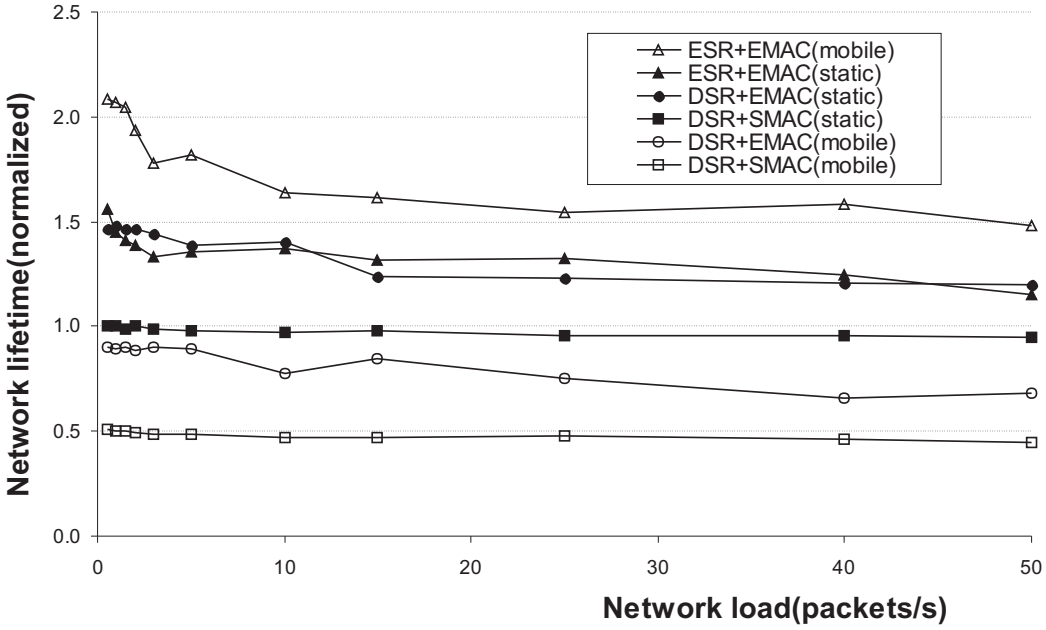


Figure 4.12: Network lifetime of two different schemes

we would expect that the lifetime of the network gets larger to some extent when the message frequency is high, due to the fact that neighboring nodes of the transmitter and receiving nodes will switch their transceiver to standby to prevent energy-waste in overhearing.

Our cross-layered approach performs better in the scenarios where the nodes are mobile than in static scenarios. This can be explained by the fact that the roles *active* and *passive* are not changed in the latter case, while in the mobile case the dynamic changes in network topology force the nodes to reconsider their role. This leads to a better and more even energy consumption between the nodes, resulting in a longer network lifetime.

Opposite to our protocols, SMAC and DSR perform better in the static case than in the mobile case. This is clearly due to the overhead in routing; in the static scenario, routes have be to established *only once*, while in the mobile scenario routes are regularly updated.

4.7 ESR Implementation and evaluation

According to the ESR simulation results in the previous section, this novel on-demand routing protocol achieves better performance over conventional routing algorithms for WSN, especially when the density or the size of the network increases. However, the simulation not guarantee that ESR works well in real network operation, because the code in a simulator is contained within a single logical component, which is simply defined and accessible. On the other hand, implementation requires the use of a system with many components and complex interactions. Furthermore, since WSN routing protocols are significantly different from traditional ad hoc routing protocols, a new set of features must be introduced to support the routing protocol. Therefore, creating a real implementation is more difficult than a simulation. In the implementation, we use not only a novel architecture and operation system to reduce energy consumption, but also use wireless sensor node prototypes and some software tools.

4.7.1 Implementation environment

Data centric architecture

A data-centric architecture organizes the internal software and hardware components of the sensor nodes of a WSN in a manner that allows them to work properly and be able to adapt dynamically to new environments, requirements and applications [21].

The data centric architecture places all the protocols around a central nucleus to dynamically reconfigure the protocols in use. This nucleus can enable or disable certain functionalities of each layer and also select, enable or disable layers during run-time, rather than cover all worse possible cases.

AmbientRT operating system

AmbientRT is a Real Time Operating System (RTOS) that fits inside the limited memory of a sensor node while still supporting low-power modes and causing only minimal overhead. Specifically, AmbientRT has very powerful features like (soft) real-time scheduling, dynamic memory allocation, online reconfigurability and support for a data driven architecture [91].

The AmbientRT kernel uses a lightweight preemptive real-time scheduling algorithm, which is theoretically proven to be deadlock free and provides soft real-time guarantees for the chosen set of tasks. To save memory AmbientRT supports dynamic memory allocation to reserve and free up memory space of arbitrary size in

a dedicated heap area. With online reconfiguration support, the AmbientRT kernel can also load and run modules dynamically. In this way, AmbientRT is able to support reconfiguration based on functionality becoming available even after device deployment. Moreover, in the AmbientRT Data-Centric Architecture is supported with its Data Manager Component. This enables its functionality to create the most efficient configuration for the changing environmental conditions, instead of offering it only at compile time.

Sensor node prototype

In the implementation, a sensor node has been used to demonstrate the effectiveness of our energy efficient routing protocols(Figure 4.13 on page 103. The hardware design consists of a small and energy efficient processor (MSP430F149 [28]), a single channel transceiver (TR1001 [29]) and additional components, including some LED's, serial memory and debugging interface. For programming the CPU, this hardware has a JTAG interface and the power is supplied by two AA Alkaline batteries.

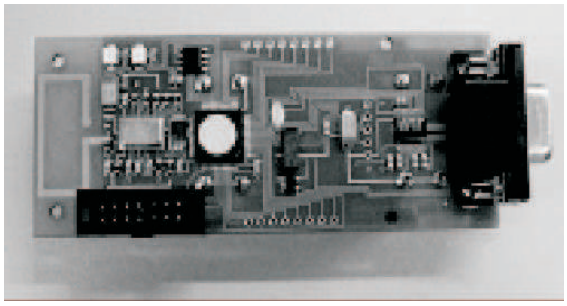


Figure 4.13: Sensor node prototype

The processor used in the EYES sensor node is a MSP-430F149, produced by Texas Instruments. It is a 16-bit processor and has 60 KBytes of programmable flash memory and 2 KBytes of Random Access Memory (RAM). The processor provides an interface to sensors, since it has an integrated analog to digital converter and it can read (or control) many I/O lines. When running at full speed (5 MHz), the processor consumes approximately 1.5 mW, but it also has several power saving modes.

The communication function between nodes is realized by a RFM TR1001 hybrid radio transceiver. It has low power consumption and a small size. The TR1001 supports transmission rates up to 115.2 Kbps. This single channel transceiver has three operational states: transmit, receive and standby. Table 4.3 on page 99 sum-

marizes the power consumption of the transceiver in three states. Typically, transmitting consumes more power than receiving and standby [97].

4.7.2 Implementation and experiment

Tasks and timing

Table 4.4: Contents of the task in AmbientRT OS

Parameters	Definitions
name	A global identifier for the task
cname	The function name of the task in the C source
deadline	The time relative to the moment the task is selected for scheduling, before which the task should finish. The deadline of a task i is indicated by D_i
period	The time between every activation of a task. The period of a task i is indicated by T_i
cputime	The worst-case computation time of a task. The CPU time of a task i is indicated by C_i
reserved	A list of resources the task uses.

With the modular support of the AmbientRT operating system, writing applications means writing tasks. Tasks are just C-functions of a predefined type. They can be identified to the OS by a Data Specification File (DSF), which not only makes them more readable and writable, but also prevents implementation mistakes and simplifies the implementation process. Tasks have a lot of parameters of their own. Moreover, a task normally needs exclusive or shared access to resources to produce or consume data. For the kernel, it is necessary to know which kind of resource and data types are used. Additionally, a task can be subscribed by a trigger. The definition of a task within DSF is shown in Table 4.4 on page 104.

Table 4.5: Two real-time tasks in LMAC and ESR implementation

name	deadline	period	cputime	resource	subscribe	dataspec
mac_timer	500	1024	200	*RADIO	TIMER0	
mac_in	500	1024	200	*RADIO	RADIOIN	*radio_message

In the implementation of ESR, there are two main real-time tasks as shown in Table 4.5 on page 104. The first task *mac_timer* is subscribed to a trigger TIMER0 and has exclusive access to the resource RADIO. It is ready to run at the beginning of a new time slot in LMAC. Each node in a WSN calls it periodically. The second task *mac_in* is subscribed to a trigger RADIOIN, with exclusive access to the resource RADIO. Moreover, it needs to read and write the data type *radio_message*. Because the task is aperiodic, each node in a WSN calls it when the radio transceiver switches on.

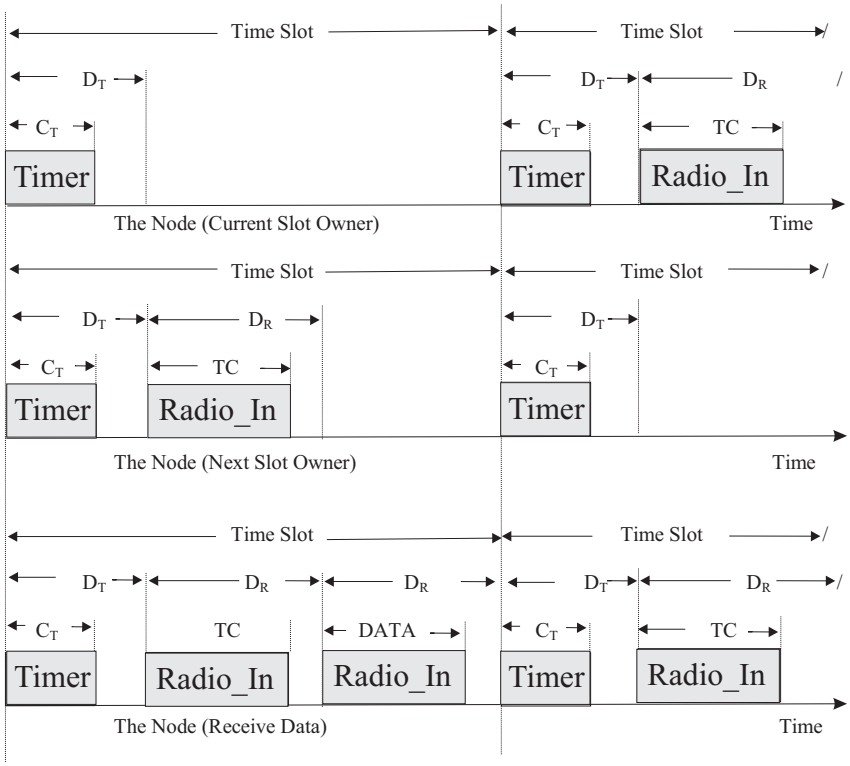


Figure 4.14: Real-time task timing operations for nodes in LMAC and ESR

In the time diagram of the implementation, as shown in Figure 4.14 on page 105, for the node that owns the current slot, it transmits TC and data messages in *mac_timer*, and then switches off the radio transceiver; *mac_in* doesn't need to be released; On the other hand, the one-hop nodes that don't own the current slot, they switch on the radio transceivers and trigger *mac_in* to receive TC message after *mac_timer*. If they are addressed in the message, they keep the radio transceiver on and retrigger *mac_in* to receive data message in the current time slot.

Further, all of ESR operations are implemented in *mac_in*. This task is used for receiving TC and data messages, which have a close relationship with ESR, such

as providing local topology knowledge and incoming data. Routing messages in ESR also need to be received through the use of the radio transceiver in *mac_in*. Hence, it makes a clear framework between real-time tasks and network protocols. Correspondingly, ESR takes charge of publishing data in the terminal instead of LMAC, and the deadline of *mac_in* is also adjusted to a suitable value.

Implementation architecture

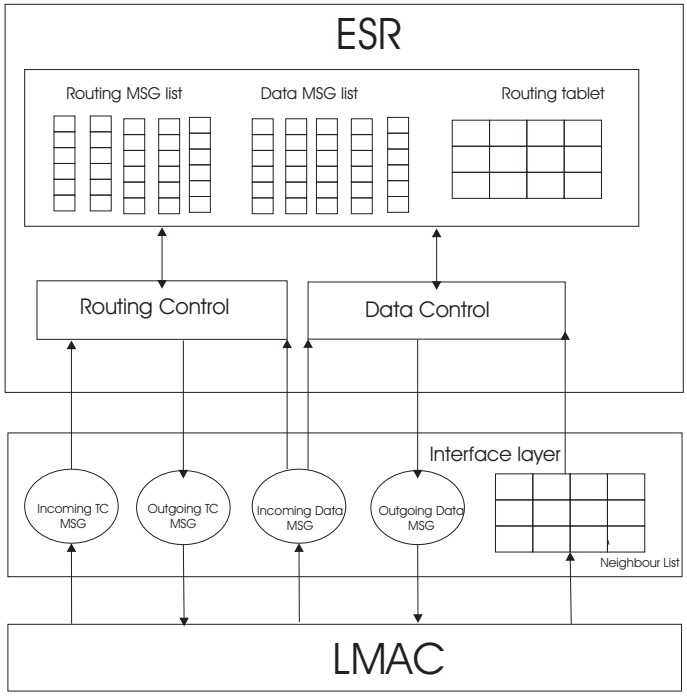


Figure 4.15: The implementation structure of ESR and LMAC

One important feature of ESR is working collaboratively with LMAC to minimize the transmission energy in the flooding of route request messages. Moreover, the ESR re-catch and re-cut process always relies on the local topology knowledge by using the LMAC neighbor table to counter the mobility and unreliability of nodes. In order to make the interaction clear and easily adapt to different applications, we designed an interface layer between the two protocols. Figure 4.15 on page 106 illustrates the whole layered view.

Specifically, a node always stores received messages from LMAC in the interface layer and then the corresponding ESR process can be triggered by the interface flag. We have implemented several special processes to handle routing messages and data messages, respectively. On the other hand, the node prepares messages in ESR for

its own slot, and also stores them in the interface layer before the LMAC process responds and transmits them in the TC and data section. In addition, the neighbor table of each node is also kept in the interface layer in order to provide local topology information for ESR. Especially, the loss of best neighbor to the source or destination is noticed by ESR immediately. Then a fast and efficient re-catch mechanism can be triggered to recover the broken links.

In the three routing phases of the ESR protocol, each node in the network has its own role, which is defined as the routing state. In each state, a node responds to a network event by different processes. In total, we identify six routing states as shown in the Table 4.6 on page 107.

Table 4.6: Routing states of sensor node in the network

State type	Definitions
request/re-request	Discovering the best neighbor in direction to the destination node in the route setup or route re-establishment phase.
on-route	Having both best neighbors in the route table and forwarding data.
re-catch	Discovering the best neighbor to an on-route node in the route maintenance.
re-cut	Find out the redundant link in a route during route maintenance.
old on-route	After receiving the error message when re-catch fails
off-route	No corresponding source-destination entry in the route table

From the view of the overall ESR routing processes, we analyzed the characteristic of each route state and transitions among them. Normally, the transition between different routing states arises from certain network events, which are normally identified by a special kind of routing message or timer. Figure 4.16 on page 108 illustrates the relationship and transition among different route states.

Experimental network setup

In order to evaluate the performance of our routing protocol, a sizable network of sensors is built to form an autonomous wireless network. For implementing an ESR algorithm on sensor node prototypes, a LCD display is attached to the sensor node to better observe the operation of an individual sensor node as shown in Figure 4.17 on page 109. Much static state information of a node can be observed through the LCD, such as the ID of the node, its time slot, one-hop neighbors ID, the current

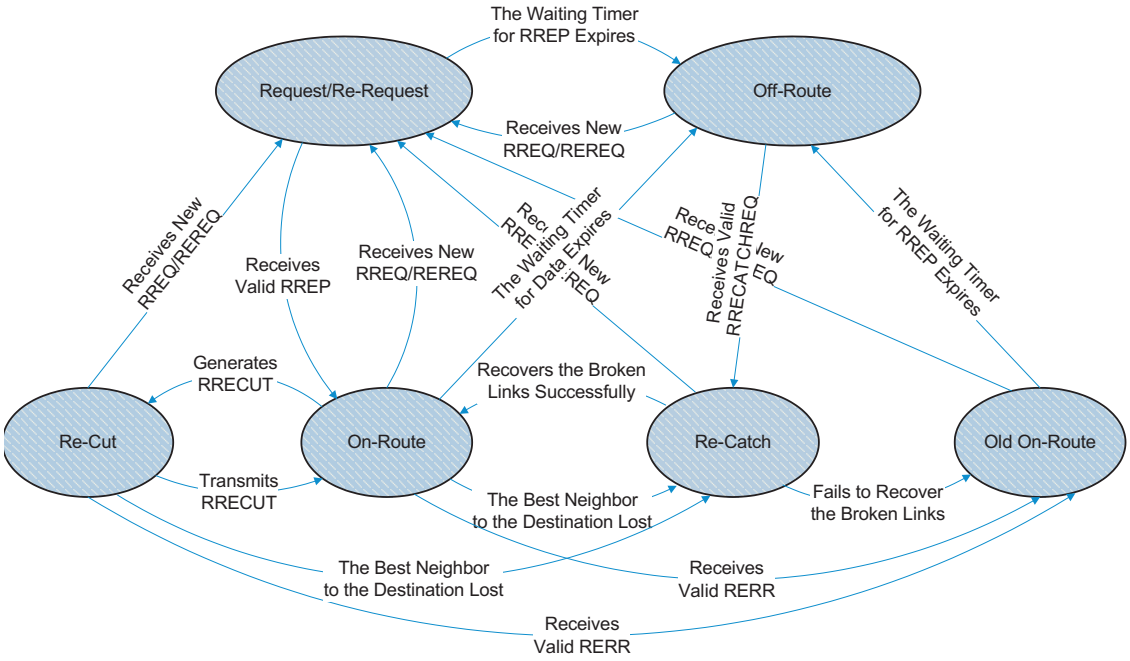


Figure 4.16: Relationships and Transition among ESR Routing States

number in the data list and routing message list, route entry records, and all records of the above quantitative criteria. When a data message arrives at the destination node, its contents can also be shown in the LCD, like the source ID, data message ID, elapsed hops number and delay time. In particular, all route entries in the LCD can reflect the changing progress when different ESR operations happen.

Moreover, several code fragments were added to capture various performance related data in the sensor network. One of the sensor nodes performs as a gateway of the sensor network to the hyper-terminal of an external computer by the connection of a serial cable. At the end of each experiment, the gateway node gathers all the performance related data from the sensor network and transmits it to the external computer, where the data is stored and displayed.

In the experiment, a sensor network with 12 nodes is built in a library hall. Due to the limitation of space, we control the radio range of nodes to about 4m so that these sensor nodes can be initially scattered randomly within a maximal square flat space of 12×12 m2. After testing the ESR performance successfully for one route in the small amount of nodes, we increased the network size further and created multiple routes from different sources to destinations. Table 4.7 on page 109 lists all the whole network environment parameters. In such a network, we set two routes among three of the nodes. One of them is the source, as well as, the destination. The hops distance from the source to the destination changes with the

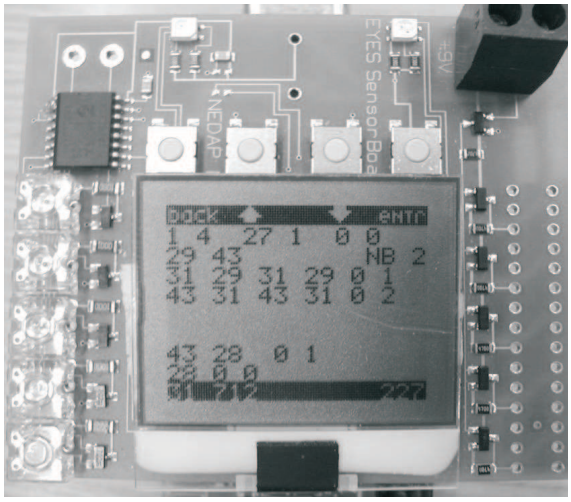


Figure 4.17: LCD device attached on the sensor node prototype

Table 4.7: Experimental network environment parameters

Maximal network size	12
Maximal network connectivity	8
Maximal network square	12×12 m2
Fixed transmission range	about 4m
Maximal source-destination pair	2
Maximal network hops	3
Node movement speed	0.5m/s

dynamic network topology. The maximal distance is about 3 hops. All the sensors in the network have a speed of 0.5m/s to a random location. After a random waiting time between [0,60] seconds, the sensor node moves again. During the evaluation of ESR performance, the number of nodes, the size of the square and the movement speed of nodes are different in each scenario. Additionally, all nodes were placed on the table with the same height of 0.8m and the experiments were done during daytime at indoor temperature.

4.7.3 Evaluation and discussion

In order to evaluate the suitability and performance of our routing protocol for WSNs, the following quantitative metrics are used in the measurement:

Two extra internal performance metrics are used in the measurement, in addition to the four metrics already used in the simulation (described in section 4.6.1 on page 93) :

- *Re-Catch Success Ratio* The ratio between the number of successful route recoveries of broken links by *Route Re-Catch* and the total number of *Route Re-Catch* attempts.
- *Re-establish Success Ratio* - The ratio between the number of successful re-establishments of broken links by *Route Re-request* and the total number of all the *Route Re-request* attempts.

The first two metrics indicate the "internal" performance of the protocol and the rest of the metrics are the same as the metrics already used in the simulation(described in section 4.6.1 on page 93) :

In the experiment, we also compare ESR with the AODV routing protocol in respect of route acquisition time, end-to-end delay, routing overhead and throughput. AODV, as described in Chapter 2, is also an on-demand routing protocol, in that it discovers routes on an as needed basis via a similar route discovery process. Moreover, AODV uses a traditional routing table, one entry per destination to maintain limited routing information. In addition, AODV uses sequence numbers maintained at each destination to determine freshness of routing information and prevent routing loops. Another common feature is the maintenance of timer-based states in each node, regarding utilization of individual routing table entries, i.e., a routing table entry is expired if not recently used. However, there are some differences between both routing protocols. The hello message of AODV may be used to detect and monitor links to neighbors. When a node fails to receive several Hello messages from a neighbor, a link break is detected. Moreover, except for the destination node, any node that has a current route to the destination node can generate a RREP. The source node chooses the route with the shortest hop count for multiple RREPs. Furthermore, AODV does not have the re-catch, re-cut and re-establish mechanisms of ESR, which assists to reduce the energy consumption during data transmission.

Re-Catch and Re-request Success Ratio

A significant characteristic of ESR is that a fast re-catch mechanism is able to recover the broken link locally and efficiently, which suppresses the rate of energy consuming route re-establishment to a minimal level. Even when route re-catch fails, the directional and geographically limited re-request flooding could re-establish

the error route with fewer number of transmissions. Hence, the Re-Catch Success Ratio (RCSR) and Re-request Success Ratio (RRSR) are important performance criterions of ESR. In the experiment, we evaluate them by the change of topological rate, network size, and network connectivity.

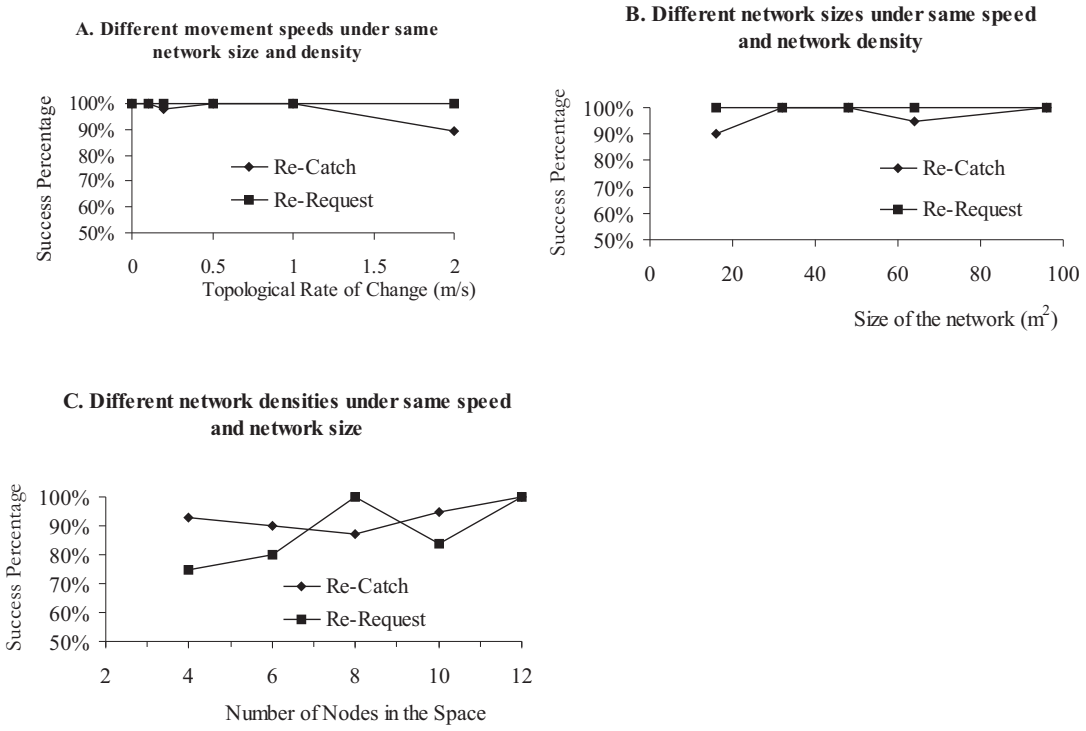


Figure 4.18: Re-Catch and Re-request Success Ratio of ESR under different experiments

In the first experiment, as shown in Figure 4.18 A on page 111, we keep the same network size and network connectivity, and evaluate RCSR and RRSR by the change of topological rate. Specifically, 12 sensor nodes are initially placed within a space of 12×8 m2 evenly with the same node density. The hops distance from the source to the destination changes with the dynamic topology. The result shows that ESR has good RRSR when the change of topological rate is high. Specifically, when the movement speed of target node is relatively low, the probability of a re-catch process occurrence in the network is also less. Moreover, once the re-catch phase occurs, ESR has enough time for nodes on two sides of a broken link to respond to the case of the loss of neighbor, which can effectively suppress the rate of re-establishment. Furthermore, the re-request success ratio is close to 100%. When the topological rate changes a lot, the routes are always in an unstable state, and the loss of neighbor often occurs. Additionally the radio transceiver of the target

node can not transmit and receive messages effectively due to too fast movement, hence the re-catch success ratio decreases. Once the re-catch fails, the process of re-establishment begins working for a new route discovery. In this scenario, the re-establish success rate keeps perfect results.

In the second experiment as shown in Figure 4.18 B on page 111, we keep the same network connectivity and topological rate of change, and evaluate RCSR and RRSR by the change of network size. Different from the last experiment, the number of network nodes and the area of space are changing, and each node always keeps moving with a speed of 0.5m/s. The results show that ESR has a good RRSR as the network size increases. Specifically, when the number of network nodes is relatively low, there is little chance to trigger the re-catch process because the route can work well in one or two hops distance. Meanwhile, re-establishment phases seldom occur, so RRSR is close to 100%. When the number of network nodes and the area of space increases, the hops distance from the source to the destination can possibly attain the maximum. The chance for the loss of neighbor also becomes high. In this situation, re-catch does work, and recovers the broken link within two neighbors due to proper movement speed. The only reason for re-catch failure still arises from radio unreliability. Like the last experiment, once re-catch fails, the re-establish performs well and establishes a new route.

In the last experiment as shown in Figure 4.18 C on page 111, we keep the same network size and topological rate of change, and evaluate RCSR and RRSR by the change of network connectivity. Different from the last experiment, the number of network nodes within a space (12×8 m2) is changing. The result shows that ESR has significantly good RCSR, especially when the network is connected. Meanwhile, the re-establishment phase seldom occurs and its success rate is close to 100%. However, when the number of network nodes in the space is relatively low, the re-catch process may occur more frequently because of lost neighbors. Moreover, the RRECATCHREQ of the target node possibly can not arrive at the on-route node because too few intermediate nodes exist in the network. This situation also influences the success rate of re-establish process. Once re-catch fails, re-request can not reestablish the route, either. Hence, re-request success ratio becomes low. When the node density increases, the chance to find that on-route node becomes higher. The reason of re-catch failures still arises from radio unreliability.

End-to-end delay

End-to-end delay (EEDD) is the time elapsed when a data message travels from the source to the destination. It is the sum of the transmission, propagation, processing and queuing delay experienced by the data at every intermediate node of the network. EEDD performance depends on multiple factors, such as the size of the

data message, route acquisitions time, the rate of topological change, traffic load of the network and data generation rate. In the experiment, we compare ESR with AODV, by the change of topological rate, network size, and network connectivity.

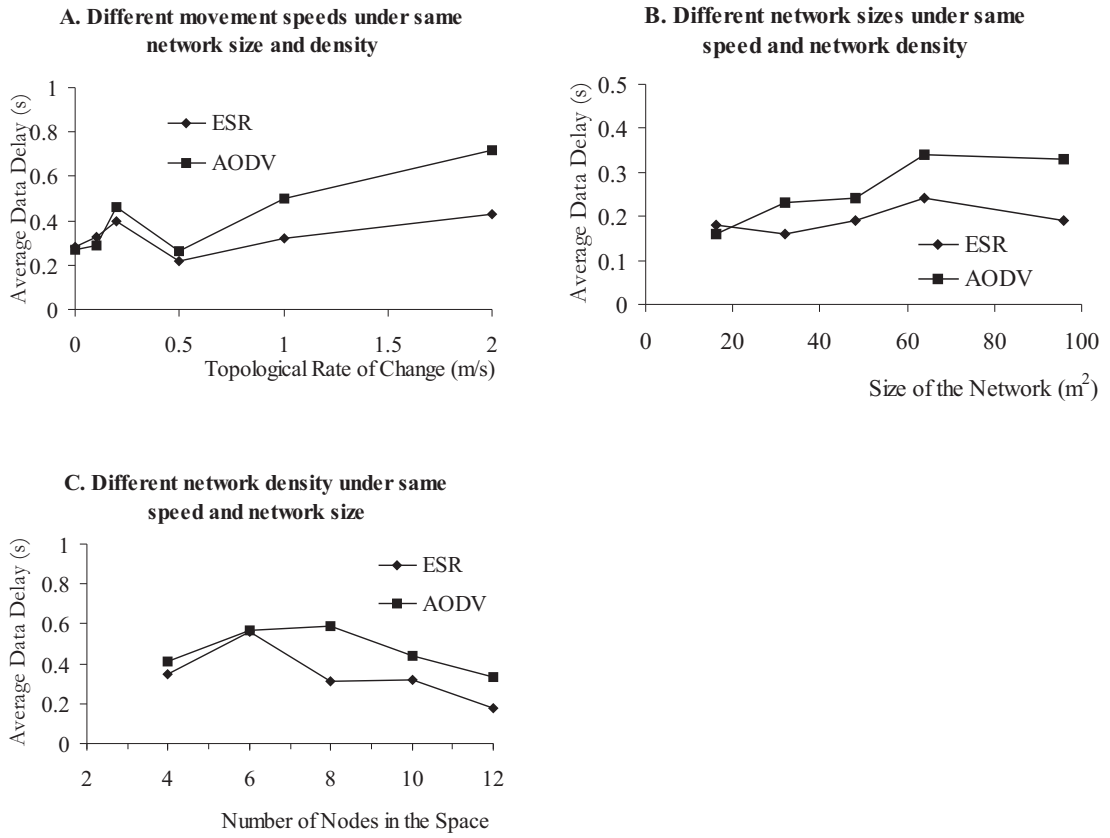


Figure 4.19: End-to-end delay of ESR and AODV in different experiments

In the first experiment as shown in Figure 4.19 A on page 113, we keep the same network size and network connectivity, and evaluate end-to-end delay by the change of topological rate. Specifically, 12 sensor nodes are initially placed within a space of 12×8 m2 evenly with the same node density. The hops distance from the source to the destination changes with the dynamic topology. The results show that ESR has better end-to-end delay than AODV, especially when the rate of topological change increases. When the movement speed of nodes is relatively low, the probability and rate of link breaking in the network is also low. So data message can arrive at the destination with little delay. When the speed of sensor movement increases, re-catch mechanism will be often triggered because the routes are always in an unstable state, and loss of neighbor occurs. As we analyze the re-catch success ratio, ESR can recover the broken link successfully and locally within a short time.

Hence unsent data message will not be in queue for too long while waiting for the local recovery. For AODV on the other hand the costs of elapsed time focuses on the transmission of route error messages and new route discovery with flooding in the whole network when a link is broken, which leads to longer end-to-end data delay.

In the second experiment as shown in Figure 4.19 B on page 113, we keep the same network connectivity and topological rate of change, and evaluate EEDD by the change of network size. Different from the last experiment, the number of network nodes and the area of space are changing, and each node always keeps moving with a speed of 0.5m/s. The result shows that ESR has better EEDD than AODV, especially when the network size increases. When the size of the network is relatively small, as we analyze in RRSR, the hop distance of route is always one or two hops and the rate of loss of neighbor is low. Hence, EEDD of both protocols maintains good performance. When the number of network nodes and the area of experiment increase, the hop distance from the source to the destination also increases. Likewise, as the situation in RRSR, re-catch of ESR can work well and recover the broken link within two hops locally. Hence, unsent data messages will not be in queue for too long time. On the other hand, AODV may spend longer time waiting on the transmission for route error messages and the new route discovery process, which results in longer data delay.

In the last experiment as shown in Figure 4.19 C on page 113, we keep the same network size and topological rate of change, and evaluate EEDD by the change of network density. Different from last experiment, the number of network nodes within a space (12×8 m2) is changing. The result shows that ESR also has better EEDD than AODV, especially when the network connectivity increases. When the density of network nodes in the space is relatively low, as we analyze in RRSR, re-catch and re-establish can not always work well due to few intermediate nodes. Hence, the end-to-end data delay of ESR is large. Likewise, AODV has a worse result compared with ESR, without re-catch and re-establish mechanisms. When the node density increases much, as the situation in RRSR, the re-catch process can recover the route efficiently and locally, with little time delay. On the other hand, AODV may spend longer time waiting on the transmission for route errors and the new route discovery process and it leads to longer data delay.

Routing overhead

A critical term of evaluating whether ESR is an energy-efficient protocol depends on the total number of routing messages (NRM) transmitted by ESR in the network. It is the sum of all control messages in the route setup, route maintenance and route reestablishment. During the implementation of ESR, we minimize the length of routing messages, and restrict their generation and transmission when the route

state changes. In the experiment, we compare ESR with AODV by the change of topological rate, network size, and network connectivity.

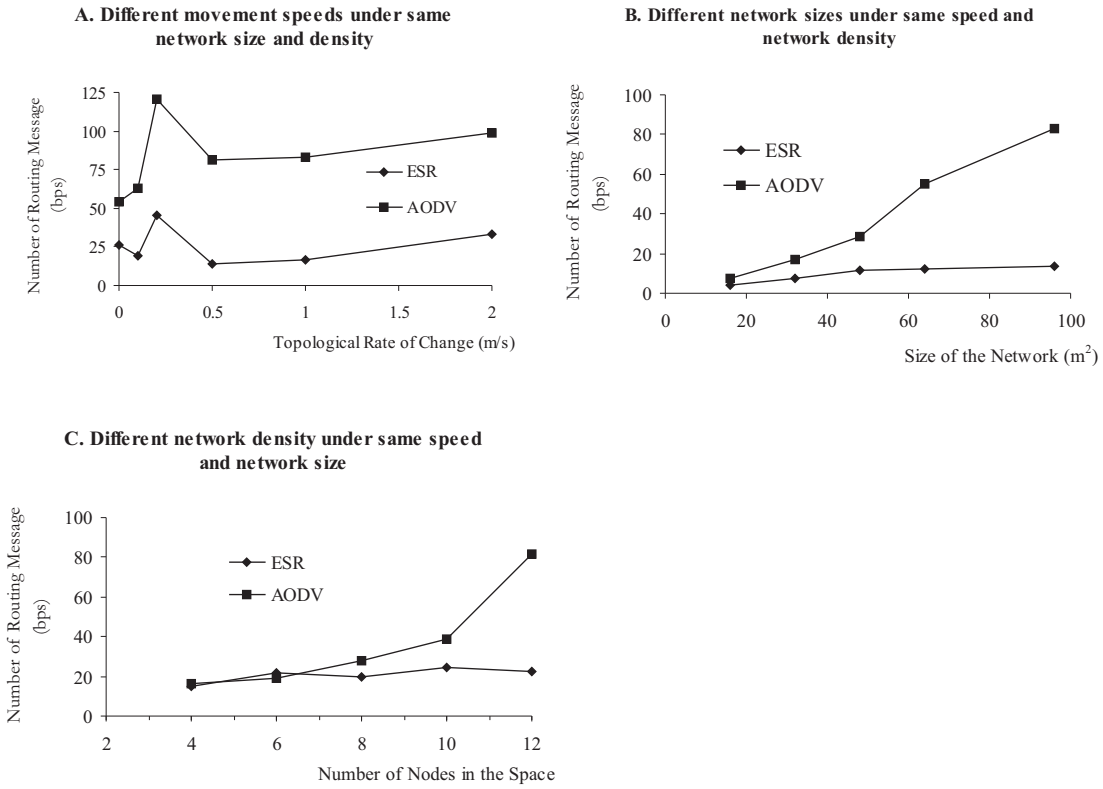


Figure 4.20: Number of routing messages of ESR and AODV under different experiments

In the first experiment as shown in Figure 4.20 A on page 115, we keep the same network size and network density, and evaluate NRM by the change of topological rate. Specifically, 12 sensor nodes are initially placed within a space of 12×8 m2 evenly with the same node density. The results show that ESR has a lower routing overhead than AODV, especially when the rate of topological change increases. As the movement speed of nodes is relatively low, the probability of the re-catch process in the network is also low. Moreover, once the re-catch phase occurs, ESR has enough time for nodes on two sides of a broken link to respond to the case of the loss of neighbor, which effectively suppresses the rate of re-establishment phase. Hence, both AODV and ESR in this situation have low control overhead. On the other hand, AODV consumes much more routing control overhead for periodical Hello messages and multiple RREP are generated, not only from the destination node. When the topological rate increases, the NRM of ESR does not decrease, obviously

due to the fact that RRSR in this situation still keeps a high success ratio. On the other hand, AODV always reinitiates the new route discovery with flooding in the whole network once a link is broken without re-catch and re-establish mechanisms, and consumes more routing control overhead through the redundant links without a re-cut mechanism.

In the second experiment as shown in Figure 4.20 B on page 115, we set the same network connectivity and topological rate of change, and evaluate NRM by the change of network size. Different from the last experiment, the number of network nodes and the area of space are changing, and each node always keeps moving with a speed of 0.5m/s. The result shows that ESR has significantly better NRM than AODV, especially when the network size increases. When the number of network nodes is relatively low, as we analyze in RRSR, the rate of loss of neighbor is also low, and the hop distance of a route is always one or two hops. Re-catch and re-establish processes seldom occur. Hence, the cost of NRM is only in the initial route setup and route header of transmitted data message. Likewise, NRM of AODV is relatively low. When the number of network nodes and the size of the network increases, the hop distance from the source to the destination increases as well. At this situation, re-catch can work well and recover the broken link locally and efficiently at low cost of NRM. Even if re-catch fails, the re-establish process, a geographically restricted flooding scheme, can realize a limited directional flooding. In addition, restrictions for generating and transmitting routing messages in the ESR implementation can also improve NRM performance to adapt to the dynamic topology in the network. On the other hand, as analyzed in the last scenario, AODV consumes too much routing control overhead in periodical Hello messages, multiple RREP is not only generated from the destination node. Furthermore, a new route discovery with flooding in the whole network will always happen when there are broken links and redundant links.

In the last experiment as shown in Figure 4.20 C on page 115, we set the same network size and topological rate of change, and evaluate NRM by the change of network connectivity. Different from the last experiment, the number of network nodes within a space (12×8 m2) changes. The result shows that ESR has significantly better NRM than AODV, especially when the network connectivity increases. When the number of network nodes in the space is relatively small, re-catch and re-establish can not always work well because few intermediate nodes exist. This situation leads to re-flooding of the network, which impacts the performance of NRM. Likewise, NRM of AODV is relatively low. When the node density increases a lot, a re-catch process can recover the route efficiently and ensures the data transmission, with low cost of NRM. Moreover, re-establishment phase seldom occurs, which avoids flooding the whole network. Hence, NRM still increases with low percentage. On the other hand, as analyzed in last scenario, AODV consumes too much routing

control overhead with the increase of more nodes in this situation.

Data throughput

Another important quantitative criterion for a routing protocol in WSNs is end-to-end data throughput (EEDT), which reflects the performance of data received in a terminal. EEDT is possibly influenced by factors like interference, multiple hop distance and channel conditions. In this experiment, we compare ESR with AODV, by the change of topological rate, network size, and network connectivity.

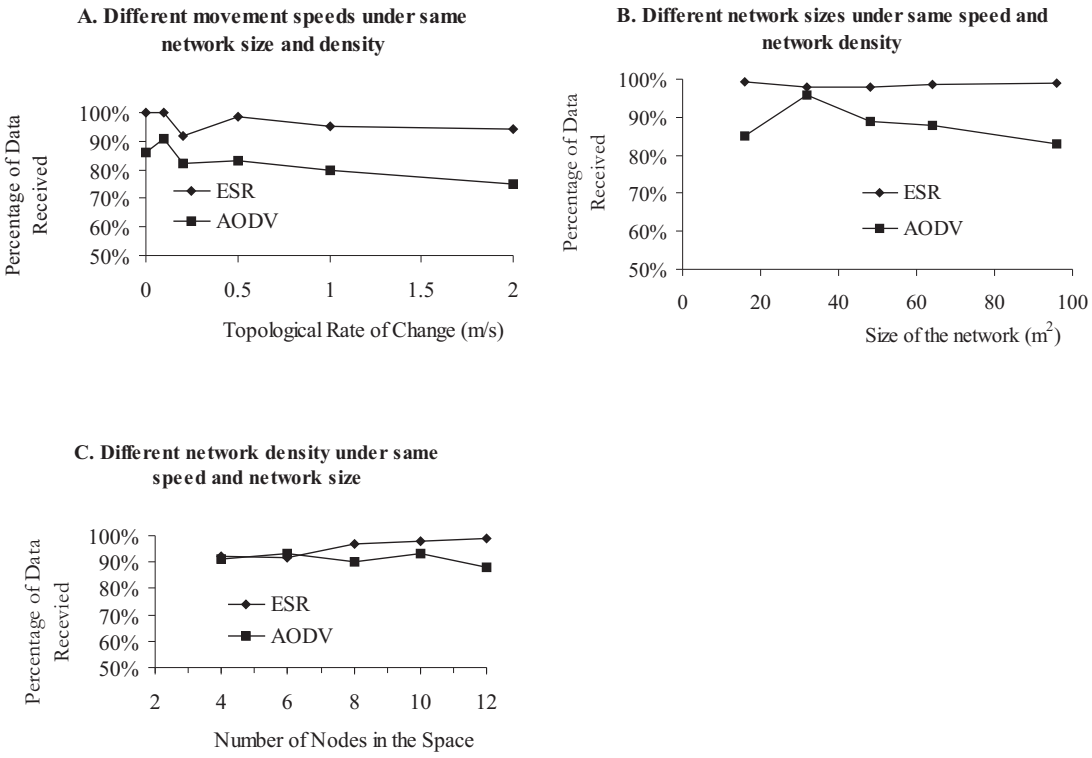


Figure 4.21: End-to-end data throughput of ESR and AODV under different experiments

In the first experiment as shown in Figure 4.21 A on page 117, we set the same network size and network connectivity, and evaluate EEDT by the change of topological rate. Specifically, 12 sensor nodes are initially placed within a space of 12×8 m2 evenly with the same node density. The result shows that ESR has better EEDT than AODV, especially when the rate of topological change increases. As the movement speed of nodes is relatively low, the probability of link errors in the

network is also less and re-catch seldom happens. Moreover, once re-catch phase occurs, ESR has enough time for nodes on two sides of a broken link to respond to the case of the loss of neighbor, which effectively suppresses the rate of re-establishment phase. Hence, EEDT of ESR in this situation has a good performance. When the topological rate changes a lot, RRSR of ESR in this situation still keeps a high success ratio above 90% although the routes are always in an unstable state, and the loss of neighbor often occurs. Furthermore, those old intermediate nodes have more chances to work for the re-established route. Thus, unsent data will not be lost often. On the other hand, AODV always reinitiates a new route discovery by flooding the whole network once a link is broken without re-catch and re-establish mechanisms. It may lead to different intermediate nodes in this route. These unsent data in old intermediate nodes will be lost, which leads to data throughput as low as 76%.

In the second experiment as shown in Figure 4.21 B on page 117, we set the same network connectivity and topological rate of change, and evaluate EEDT by the change of network size. Different from the last experiment, the number of network nodes and the area of space are changing, and each node always keeps moving with a speed of 0.5m/s. The results show that ESR has better EEDT than AODV, especially when the network size increases. When the number of network nodes is relatively low, the changes of loss of neighbor is also low, and the hop distance of route is always one or two hops. Hence, EEDT of ESR is close to 100%. When the number of network nodes and the area of space increase a lot, the hop distance from the source to the destination also increases. Likewise, as the situation in RRSR, re-catch of ESR can work well and recover the broken link within two hops locally due to proper movement speed. Furthermore, each node has a data cache to store unsent data, and Time To Live (TTL) field of the data can be updated to live longer in the data list when the route is being restored locally. Hence, EEDT of ESR in this situation also has good result. On the other hand, AODV may lose more data messages because of a longer waiting time spent on the transmission for RERR and a new route discovery process. Moreover, once the old route node is not the intermediate node in a new route any more, its unsent data messages will be lost.

In the last experiment as shown in Figure 4.21 C on page 117, we set the same network size and topological rate of change, and evaluate EEDT by the change of network connectivity. Different from the last experiment, the number of network nodes within a space (12×8 m2) is changing. The results show that ESR has better EEDT than AODV, especially when the network connectivity increases. When the number of network nodes in the space is relatively low, as we analyze RRSR, re-catch and re-establish can not always work well because few intermediate nodes exist. Hence, EEDT of ESR is not perfect. Likewise, AODV has similar results

to ESR, without the re-catch and re-establish mechanism. When the node density increases, like the situation in RRSR, the re-catch process can recover the route efficiently and ensures the data transmission. Furthermore, the useful assistant of ESR implementation, as analyzed in the last scenario, cause EEDT of ESR to have a better performance than AODV.

Although our real world implementation and testbed setup are different from the simulation setup, from the test results we can see the same performance trends in both cases, which is that our cross-layer approach has higher reliability and less control overhead in the dynamic network setup.

4.8 Conclusions

Our lessons learned in developing network protocols for wireless sensor networks in the last couple of years show that using the traditional layered networking approach has several drawbacks in the resulting performance and efficiency of the system. Quite often, significant improvements are possible for network protocols. In this document we proved that cross-layer optimization is a useful approach for WSNs.

We discussed a TDMA-based medium access protocol, which operation is not dependent on a central manager or base stations. The nodes in the network are capable of choosing their own time slot, based upon local information only. Nodes in the network can communicate with each other collision-free. Not every node is needed to actively participate in communication in the network for global connectivity. Hence the medium access protocol allows some nodes to be *passive*. These passive nodes save energy by not controlling a time slot, but make use of a backbone in the network that is formed by active nodes. In this way, the medium access protocol overhead is greatly reduced for passive nodes. Passive nodes can communicate with active nodes, although this communication is not guaranteed to be collision-free.

We also introduce a new on-demand routing protocol EYES Source Routing (ESR), which benefits from local topology information that is already present in the medium access protocol. In ESR, limited routing information stored in the intermediate sensor node reduces the amount of energy spent on data transmission. A fast recovery mechanism relying on the MAC layer feed back is implemented to counter the mobility and unreliability of nodes. In the route reestablishment, a directional and geographically restricted flooding scheme based on the previous knowledge of the location of destination node is devised, which uses Hops To Live (HTL) field in the request to realize a limited directional flooding.

We compared our cross-layer optimized networking protocols with traditional

protocols for WSNs: SMAC and DSR. One of the key issues in WSNs is the network lifetime. Simulations show that in equal network configurations, message frequency and size assumptions, our cross-layered approach demonstrates a better network lifetime, especially in the case where nodes are mobile. In a static case, the difference is smaller, mainly due to the fact that the routing protocols have to establish a route only once. Our protocol has standard a small amount of data reserved for route updates; in the static case this space is wasted.

We also implemented the proposed protocol combination in a real-life testbed. The results show that our protocols achieve improved performance on route acquisition time, end-to-end data throughput and delay when the density or the size of the network increases. Moreover, the performance gains become more significant when the topological rate of change is relatively high. Consequently, ESR can reduce energy consumption and handle the dynamic topology of WSNs in the real-world environment

Chapter 5

Cost-based data-centric routing for WSN

The resource limitations of wireless sensor networks (WSN), especially in terms of energy, require a novel and collaborative approach for wireless communication. In recent years, data-centric technologies have been proposed to perform in-network aggregation of data to yield energy-efficient dissemination in WSNs. However, not much work in the data-centric routing domain has been done in relation to the dynamics of wireless sensor networks. In this chapter, we focus on the dynamic aspects and present a new reliable cost-based, data-centric routing algorithm for such dynamic WSNs. We address the reliability of dynamic wireless sensor networks in the point-multipoint routing scenarios by a data-centric approach. Current research in this area generally assumes a rather static network, leading to a strong performance degradation in a dynamic environment. A network wide reflooding of messages is the common solution to network topology changes. The situation gets worse when the data sink moves and a stable network is hardly possible to form. In our research we try to maintain the communication when sensors move, such that less energy is used to re-setup the network. Moreover, event mobility poses a great pressure on the WSNs. We designed the routing protocol to route message intelligently to reduce the effect of event mobility. The work presented in this chapter has been published in SNPD'06 [108][1].

[1]Jian Wu and Paul Havinga. "*Reliable Cost-based Data-centric Routing Protocol for Wireless Sensor Networks*". In Proceedings of 7th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, Las Vegas, USA, 2006.

5.1 Introduction

There are many challenges in *wireless sensor networks* (WSNs). One of the key tasks of a WSN is to deliver sensed data to interested parties who requested it. As these networks are expected to be densely deployed, a major issue in the study of networking protocols for sensor networks is related to data dissemination and routing. A vast literature exists on routing protocols for ad hoc networks (MANETs) as described in Chapter 2. However, sensor networks differ from ad hoc networks in many ways, including data redundancy, type of information to be exchanged, capability of nodes, amount of traffic and application scenarios of interest. Therefore, even though strictly speaking a sensor network is an ad hoc network, routing schemes devised for ad hoc networks do not directly apply to this case. While many of the ideas found in the aforementioned literature can be used to design adequate routing schemes for sensor networks, it is likely that novel schemes will have to be designed in order to suit the specific requirements of these networks.

In recent years, a number of papers have appeared which present solutions relevant to the issues of routing and data dissemination in sensor networks. In these works, *Data aggregation* has been highlighted as a particularly useful paradigm for routing in wireless sensor networks [31] [33]. It attempts to collect useful information from the source sensors surrounding the event. It then combines the data and transmits only the useful information to the data sink thereby eliminating redundancy, minimizing the number of transmissions and thus saving energy. This paradigm shifts the focus from the traditional *address-centric* approaches, which try to find short routes between pairs of addressable end-nodes, to a more *data-centric* approach, which focus on finding routes from multiple sources to a single data sink. This allows in-network consolidation of redundant data.

In [40], Intanagonwiwat et al. proposed a popular data aggregation paradigm for WSNs called directed diffusion. Directed diffusion is a data-centric and application aware paradigm in the sense that all data generated by sensor nodes is named by attribute-value pairs. Moreover Schurgers et al. [84] proposed another variant of directed diffusion, called Gradient-Based Routing (GBR). The key idea in GBR is to memorize the number of hops when the interest is diffused through the whole network. However, in both works the dynamics of the network is almost neglected, which poses a great challenge in the dynamic network environment. Any topology change in the network requires a network wide flooding of messages to reset-up the gradient field. More recently, GRAB (GRAdient Broadcast) [26] protocol has been proposed. The basic idea is that data packets issued by a sensor will be delivered along the direction of a sink by descending some cost, which is initially built and maintained by the sink but kept by each sensor. GRAB requires each node's cost value to be periodically refreshed by a sink initiated flooding, causing the same

problem of excessive overhead under the assumption of a dynamic network.

This chapter presents a Reliable, Cost-based, Data-centric Routing protocol (RCDR) for wireless sensor networks as part of the European research project EYES [25]. In this work, we address, in particular, energy efficiency and the dynamics of wireless sensor networks. Instead of each sensor sending its own data report directly to the data sink, we introduce a global-local gradient paradigm that only sends the aggregated data from the center of the event to the data sink. This ensures that sensors aggregate the collected data as close as possible to its origin, while only a small number of aggregated data are sent to the data sink. To increase the reliability of these aggregated data, they are sent via multiple adjustable routes to the data sink. Local algorithms are designed to resume the network gradient when the network topology changes, especially the mobility of the data sink is solved with a negative gradient. Furthermore, the movement of a sensed event, such as in the event tracking applications, is also efficiently dealt with.

In Section 5.2 we give an overview of the characteristics of the wireless sensor networks and our design goals for the routing protocol. Section 5.3 discusses the design of the Reliable Cost-based Data-centric Routing protocol (RCDR), that is especially designed for mobile WSNs and that exploits the benefits of the data aggregation approach discussed in this chapter. Section 5.4 discusses our simulation results and finally Section 5.5 gives a conclusion and area of future work.

5.2 Problem Statement

5.2.1 Wireless Sensor Network characteristics

Habitat and environmental monitoring represent a class of sensor network applications with enormous potential benefits for scientific communities and society as a whole. This chapter will focus on the routing algorithm for WSN in this application domain. The characteristics of such wireless sensor networks are:

- **Data Sink.** In the network, only a small number of nodes are data sinks. They constantly probe the network for certain data and collect responses from the network.
- **Event Source.** As monitored or tracked events appear in the network, the sensor nodes detect these events and generate data reports according to their sensor readings. Multiple event sources could exist simultaneously at different parts of the network. The type of monitored event depends on the query sent out by the sink.

- **Data Flow.** Once an event appears in a certain area of the network, a constant and continuous data report will be generated by detecting sensors. The data flow lasts until the event disappears.

- **Mobility.** The sensor nodes and data sinks generally work in a static manner. However, in many applications slow movements of sensors and sinks are expected. The frequency and speed of these movements is relatively low, compared with the data rate generated by the event source. Mobility of the event source is also considered important in our scenarios. As an event happens, it could stay in the same location, such as temperature and pressure monitoring. However, the detected events could also move in the network, such as motion detection and object tracking. When the source of the event moves, new sensors along the path will have readings and will generate a data report.

- **Geographical Information.** Obtaining geographical information is relatively expensive, considering tiny and resource limited sensor nodes. In our routing protocol, sensor nodes generally do not need geographical information when forwarding data in the network.

5.2.2 Routing Protocol Design Goals

Routing protocols should be able to transport data from the source to the destination across multihop networks. In the design of our protocol, several requirements are set up:

- **Energy Efficiency.** Because wireless sensors only carry a small battery, they have a limited energy supply. Thus low-power operation is a must. The routing protocol should be able to route data in an energy efficient way.

- **Data Centric.** In WSN, a typical mode of communication is from multiple data sources to a data sink, rather than point-to-point communication between a pair of nodes [49]. In our research, we shift the focus from the *address-centric* approach (finding short routes between pairs of addressable end-nodes) to a more *data-centric* approach (finding routes from multiple sources to a single destination based on data types, which allows in-network aggregation of redundant data).

- **Mobility.** Current data-centric routing protocols have not been able to deal with node mobility properly. A network wide reflooding of messages is the common solution to network topology changes. The situation gets worse when the data sink moves and a stable network is unable to be setup. In our research,

we try to maintain the communication locally while sensors move, so that less energy is used to re-setup the network. Moreover, event mobility put a lot of pressure on the WSN. We have designed a routing protocol to route messages intelligently, thus reducing the effect of event mobility.

- **Data aggregation.** According to the characteristics of WSN, data aggregation is a particularly useful paradigm for multihop routing. It combines the data coming from different sources, eliminates redundancy, minimizes the number of transmissions and, thus, saves energy [50].

- **Reliability.** Wireless transmission and mulithop routing causes packet losses in WSN [60]. The routing protocol should ensure that the sensing data is being transmitted to the destination successfully. The reliability of the data, especially for the aggregated data, should be enhanced as any loss is unacceptable.

5.3 Protocol Design

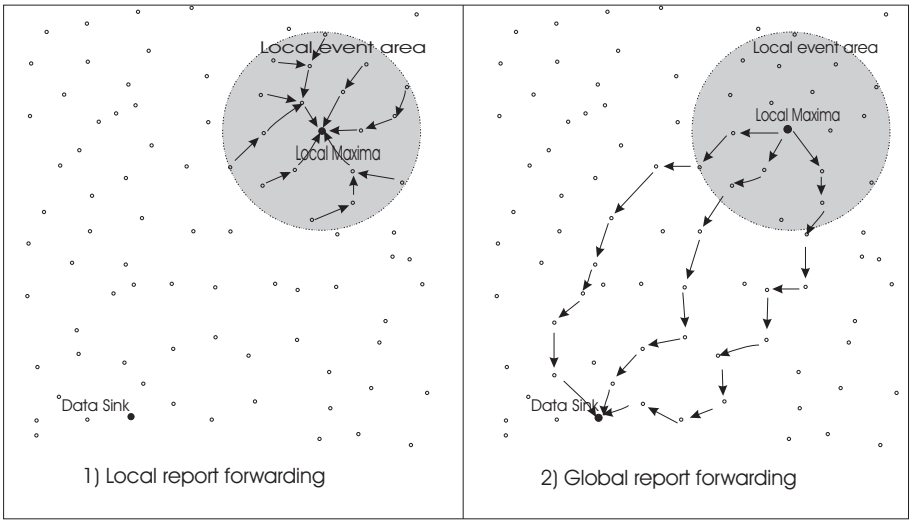


Figure 5.1: Protocol overview

5.3.1 Overview

The proposed *reliable, cost-based, data-centric routing* (RCDR) protocol has two kinds of gradients, the *global gradient* and the *local gradient*. When a data sink

wants to collect data from the network, it sends out a data query to set up a *global gradient* in the entire network. While this query message propagates in the network, each sensor establishes its own cost value toward this sink. Then any data sent towards the sink follows through the *global gradient* by multipath routing. The multipath degree is controlled by the *premium cost* of the data. *Sensor movement adjustment scheme* and *Sink movement compensation scheme* efficiently resume the disrupted *global gradient* by local interactions between sensors. Thus, reflooding, which takes a lot of energy in the network, is reduced to a minimum, while still maintaining the reliability of the network.

Locally, the footprint gradient of the event's effect is used to setup the *local gradient*, which overrides the *global gradient* in the local event area (LEA). When the sensor senses an event and sends an event report, it is first routed to the local maxima, which is the data aggregation point, as shown in the shaded area of Figure 5.1-1 on page 125. When the local maxima has processed the data from the surrounding sensors, it sends the aggregated report to the sink via the *global gradient* as shown in Figure 5.1-2 on page 125. Further more, the *Local maxima handover scheme* lets the data aggregation point moves along with the event, so that data are always aggregated locally closest to the event center.

The detailed procedure of the routing protocol is introduced in the following sections.

5.3.2 Global gradient

The gradient setup follows a similar approach to the one in [26]. However, in our work we introduce more flexibilities to the cost concept, in order to control the characteristics of data forwarding. Moreover, *waiting time* and *forwarding probability* are used to minimize the delay and reduce the broadcast storm problem. A broadcast storm happens when a message has been broadcasted across a network. This message may result in even more responses and each response results in still more responses, creating a snowball effect. This snowball effect can have a serious negative impact on network performance.

Global Cost Setup

When the data sink wants to monitor or query the network, it sends out a Data Query (DQ) message to the network, which will be rebroadcasted by the sensor nodes. The intermediate sensors not only remember the data query but also activate its sensor. From the content of the DQ message, node i also sets up a cost C_i from itself towards the data sink. Initially sensors have no information from the data

sink, so it sets its cost to be ∞. The DQ message sent out by the sink has a cost of $C_{DQ} = 0$. Each node has a link cost table to all its neighbors, which is explained in the following section. If intermediate node i receives a DQ message from its neighbor j, it adds the link cost C_{ij} to the cost of received DQ message and it sets the new cost to be $C_i = min(C_i, C_{DQ} + C_{ij})$. Meanwhile, it remembers from which node it could transfer data with the lowest cost. We call this node the Lowest Cost Neighbor (LCN). If the node receives the same DQ message from a different neighbor, it computes the value again and updates the cost and LCN node, if necessary. After a time out T_w, the node rebroadcasts the DQ message with its current C_i as the C_{DQ} and with a forwarding probability p_f. This ensures that the node only broadcasts once the same DQ message, although multiple copies could be received from different neighbors. After time out T_w, any copy of the same DQ message is ignored. After the *global gradient* setup, each node in the network should have a cost C_i and the whole network becomes a directed graph toward the sink. The node that did not receive a cost after the global gradient setup (because of collision or errors) will obtain one by the Sensor movement adjustment scheme.

The time out T_w should be sufficient to reduce the effect of broadcast storm, which results in serious redundancy, contention and collisions [70]. At the same time, it should ensure that the node obtains the minimum cost C_i in the shortest delay. If all the nodes have the same time out T_w, then neighbors will compete for the medium at the same moment causing collisions. The simple solution is to let the nodes have a random time out between $[T_{min}, T_{max}]$. This will greatly reduce the chances of collision in a collision detection based MAC layer. However, it appears to cause very large delays in the far end of the network. A further improvement is to have a random time out proportional to the node's cost C_i to the sink. The farther away from the sink, the larger the time out.

Link Cost

Each node in the network maintains a link cost table $Table_{cost}$ to all its neighbors. For each neighbor in the radio range, it has a cost value in $Table_{cost}$. The selection of cost values decides the forwarding character of this sensor network. For example, a delay related cost would make this WSN have an overall lower delay; an energy related cost would increase the life time of the WSN. The candidate schemes for Link Cost are:

- The neighbors have the same link cost C_{link}, which put all the neighbors in the same cost level. Then, the *global gradient* becomes a hop count based scheme. The message will be forwarded according to the lowest hop count routes. These routes also have a shorter delay because of the shortest path.

- The link cost to a neighbor j depends on the radio quality Q_{ij} between this node i and node j. The proposed relation could be expressed as $C_{ij} \propto \frac{1}{Q_{ij}}$. The higher the radio quality the lower the link cost. Therefore, a more reliable link has a higher chance of forwarding data in the data dissemination.
- The link cost to a neighbor j depends on the remaining energy level Er_j of node j. Every node constantly sends out its own remaining energy level. Therefore, each node builds up a table $Table_{energy}$ of its neighbor's remaining energy levels. This table can be directly interpreted into a link cost table $Table_{cost}$, if the relation $C_{ij} \propto \frac{1}{Er_j}$ exists. Such a link cost table will allow the data to go to the higher remaining energy area of the network in the data dissemination.
- a combination of above mentioned schemes could balance the performance on different aspects.

Parameters in the Forwarding

Globally a node sends its data to the sink by broadcasting the data (packet) with several forwarding parameters. The source node id and the sequence number uniquely identifies the data. Each node will only forward the first data it receives and will ignore other copies.

Each data also has a cost of its own, which consists of two parts. The *basic cost* of a data is the cost of the node which sends the data. It can be expressed as C_{basic}=C_i. This is the minimum cost the data should have in order to reach the sink from node i. The *premium cost* of a data is set to control the multipath degree of the forwarding paths. When the data has more premium cost to spend, it will travel further away from the low cost paths and go through higher cost areas. The premium cost will increase the reliability of the forwarding at the expense of more energy consumption. Thus, the overall cost of a data is expressed as $C_{data} = C_{basic} + C_{premium}$. When a intermediate node j receives new data, it compares the cost of the data C_{data} with its cost C_j.

- If $C_{data} \geq C_j + C_{ji}$, the data has enough credits to go through this node towards the sink. It will forward the data with a new cost $C_{data} = C_{data} - C_{ji}$. C_{ij} is the link cost between node j and node i.
- If $C_{data} < C_j + C_{ji}$, the data has no more credits to be forwarded. Node j will drop the data.

Since each sensor only forwards the first copy of the data, the duplicated data packet is discarded by the forwarding node even if it has enough credit. As long as

the packet can not traverse back to the same node, loops will not occur. Multiple copies of the data travel through different routes and the ones with enough costs reach the data sink. When the sink receives sensed data from the network by multiple copies, it needs a reverse query route to address the event area. Either additional queries or reinforcement can be sent to the area via the reverse query route.

Sensor Movement Adjustment

The routing protocol presented in this work is a multipath routing protocol. When the data travels in the network towards the sink, it flows through the lower cost nodes as shown in Figure 5.2 *I* on page 129. If the connections between node A and node C breaks because node C moves away from node A, the data from node A can still go through both node B and node D. If the network density is high enough, the data will bypass the troubled link and resume the reliable multipaths as shown in Figure 5.2 *II* on page 129. So, the movement of an individual sensor node does not break the data transfer in its original location area.

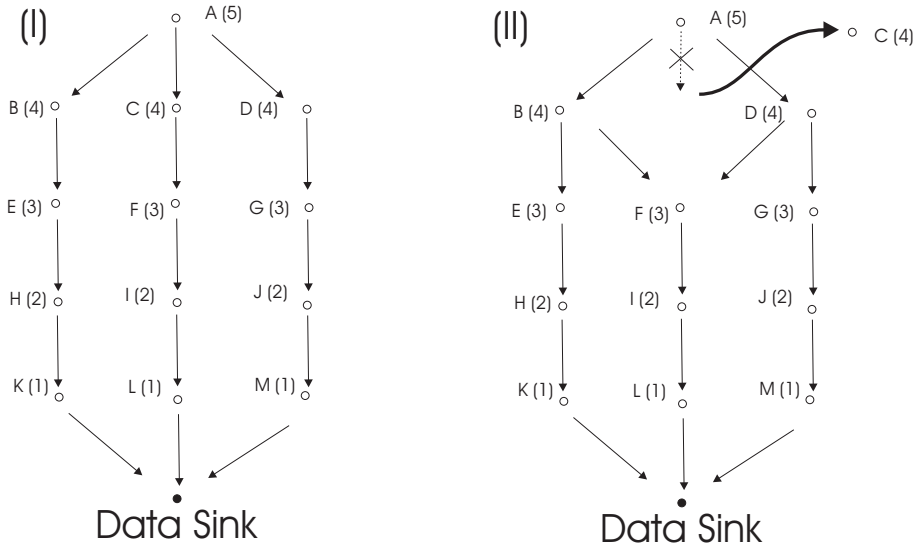


Figure 5.2: Sensor movement in a dense network

The effect of a moved sensor node in the new allocated location area is also negligible. When a node moves, it could move in two directions in respect to the sink, as shown in Figure 5.3 on page 130. If node a moves into a higher cost area, it will have the lowest cost value among its neighbors. Then, it forwards any data from its neighbors, but, in turn, its neighbors forward none of its data. If node a moves into a lower cost area, it will have the highest cost value. Then, it forwards no data from its neighbors, but its neighbors forward any of its data. In both cases,

node a is excluded from the network communications.

If only a small number of nodes in the network move, the network can still remain functional. However, if more and more nodes are excluded from the data forwarding, the network becomes very unreliable. A network wide reset is needed from the sink to restore the gradient field. However, frequently reseting consumes too much energy from the energy restrained sensor nodes. A *sensor movement adjustment scheme* is designed to minimize the effect of sensor movement by local interactions.

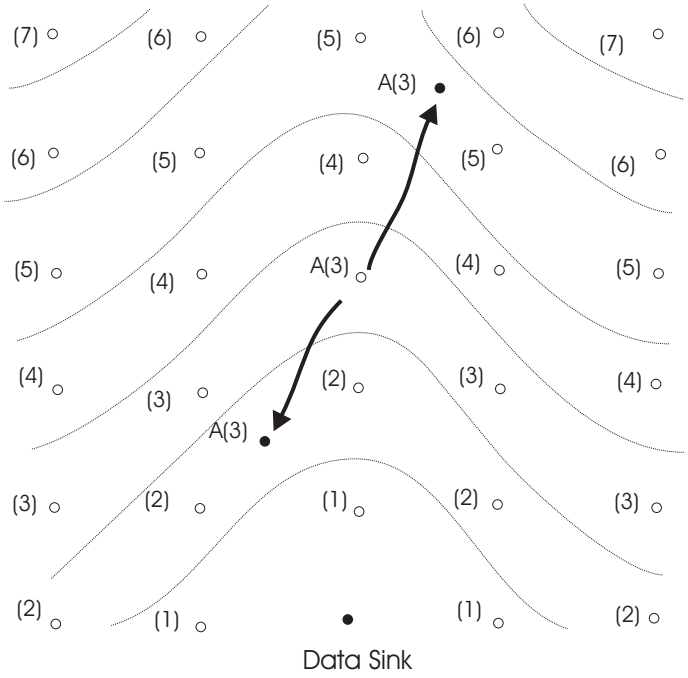


Figure 5.3: The effect of sensor movement in two directions

The movement of sensor nodes in the network can be detected by the changes of its neighbors. A cross-layer approach similar to [39] allows the node to obtain neighbor information from the MAC layer in a very efficient manner. When a new neighbor joins or an old neighbor leaves, the sensor node takes different actions to adjust its own cost value.

If a node notices that one of its neighbor has disappeared, it first checks whether this is the Lowest Cost Neighbor (LCN) node. If it is LCN node, the node sets its cost to be ∞ and sends a Cost Query (CQ) message to its neighbors. The neighbors reply with their own cost value and the node sets its cost and LCN node accordingly to the rules described in Section 5.3.2. Meanwhile, if a node receives a CQ message from its LCN node, it also sets its cost to ∞ and sends out its own CQ message with a delay. If it is not LCN node, the node simply ignores it.

If a new neighbor comes in, the node sends out a Cost Update (CU) message to each other. The node puts its own cost to the sink in the CU message. When a node receives a CU message from node j, it compares its own cost C_i with the new cost $(C_{CU} + C_{ij})$. If $C_i > (C_{CU} + C_{ij})$, then it has found a new Lowest Cost Neighbor (LCN) node and sets the new cost to be $C_i = C_{CU} + C_{ij}$.

If some neighbors are joining and leaving at the same time, the node first deals with lost neighbors and then with the new neighbors.

Sink Movement Compensation

The data sink is a data collection point in the WSN. It receives aggregated data responses from the network. In the operation of WSN, various scenarios require the data sink to be able to move in the network while collecting the sensing data. Any data loss caused by the sink movement decreases the reliability of the network. Previous work on the routing of WSN mostly focused on the static network. Particularly in the data centric route scenarios, any movement in the network will disrupt the network setups and results in data losses. When the sink moves, a network-wide broadcast is needed to restore the network gradient as in [49][36][26]. This chapter introduces a new *sink movement compensation scheme* with negative gradient, which only requires a local update in order to compensate for the sink movement.

The data sink sends out a Data Query (DQ) message to the whole network, which will be rebroadcasted by the sensor nodes. These initial DQ messages sent out by the sink have a cost C_{DQ} of 0. When the *global gradient* is set up in the network, sensing data can travel from the sensor nodes to the data sink by following the gradients. When the sink moves away from its current location, it should be able to first detect its own movement before it can carry out adjustment for the gradients.

The movement of the data sink can be detected by an additional localization mechanism. As only a limited number of sinks are needed to collect data, additional hardware on these sinks would not significantly increase the cost of the WSN. In the situation of no additional hardware or localization protocol, we have designed a simple method to detect the relative movement of the data sink. Again, the proposed method of movement detection is to monitor the changes of the sink's neighbors. When any new neighbor appears or any old neighbor leaves, we can infer that the sink has changed its location. However, this could also be the case that the sink remains static but its neighbors move away or they are simply switched on and off. We treat the second case the same as the first one and it has little effect on the sink movement compensation. The only disadvantage is that there must be more frequent local updates which involves a small number of radio transmissions.

When the sink detects its own movement (or relative movement), it sends out a negative gradient broadcast locally to compensate for its own movement. Initially DQ messages sent out by the sink have a cost C_{DQ} of 0. Immediately after it moves, it decreases C_{DQ} to -1 and broadcasts the new Degrade Update (DU) messages with a Hops-to-Live (HTL) field set to h. When the node receives the DU messages, it follows the steps described in section 5.3.2 to set its new cost. Firstly it checks the HTL field. If $HTL > 0$, it lowers its own cost and rebroadcast the DU messages with new HTL=HTL-1. The new DU message propagates until it reaches the h hops neighbors. Thus, all the sensors in this degraded area set their costs one step lower towards the new location of the sink. This creates a small funnel with negative gradient around the sink in the *global gradient* field. As a result, when the data sent by the node reaches the vicinity of the sink, it will still flow to the new allocated sink by following the "small funnel". In this way, a network-wide readjustment is avoided by only a locally restricted gradient broadcast.

The sink repeatedly decreases the cost and increases the HTL of the DU messages when it moves again. So that the DU messages can still reach the original location of the sink, the degraded area will expand accordingly. The proposed relationship between the cost and HTL is h = H - C_{DQ}, where H is a constant.

However, as the sink moves further away from its original location, the efficiency of routing data through the degraded area decreases. Firstly, the diameter of the degraded area continues to increase. More nodes are involved in the local broadcast when the sink updates the gradient cost. Secondly, the data sent back by the nodes need to travel more to reach the sink than the shortest possible path. A limit of *local gradient* compensation is set when the diameter of the degraded area gets too large compared with the network diameter. A network wide gradient reset is done by sending out new DQ messages is required to reestablish the gradient field in the network.

5.3.3 Local Gradient

Locally, we use the footprint gradient of the event's effect to route the event report. When the sensor sends an event report, it is first routed to the local maxima, which is the data aggregation point. When the local maxima has already processed the data from the surrounding sensors, it sends the aggregated report to the sink. The procedure of local maxima election and gradient setup is explained in detail in the following section:

Local Gradient Setup

A physical event leaves some footprints in the environment. For example, fire increases temperature and gas leakage increases density. Moreover, most physical phenomena follow a diffusion property with distance, i.e., $f(d) \propto \frac{1}{d^{\alpha}}$, where d is the distance from local maxima, f(d) is the magnitude of the event's effect and α is the diffusion parameter depending on the type of effect. In WSN, the sensors capture this event's effect in terms of sensed signal strength. In [27], this information gradient has been used to route the query from the data sink to the event source. However, in our research, this natural and freely available gradient has been exploited in a different way. We use it to efficiently forward the event report from the surrounding nodes to the center of the event or the local maxima.

When an event happens in the WSN, the surrounding sensors sense the event and generate readings. Because of the diffusion effect, the sensor reading is a function of distance d and time t, i.e., $R(d,t) \propto t/d^{\alpha}$. Because sensors can not detect changes if the effect is below a certain threshold, the event's effect is not infinite. After a distance D, the sensor readings becomes zero and the information gradient is not available. Therefore, we define the area inside diameter D as the *local gradient area* (LGA).

In LGA, all the sensors detect the event and generate event report R_i. After a random back-off time between [0, t_i], sensor i starts to send out the event report with its own reading R_i to the neighbors, i.e., N_i. The timer t_i has a linear inversely proportional relationship with R_i, which means the larger the reading, the smaller the time out.

After the initial round of message exchange, sensors in the LGA get a list of readings from their neighbors.

- If $R_i > \text{Max}[R_j]$ (j $\in N_i$), then sensor i becomes the local maxima of this event, i.e., $\mathbf{L}_m$. It then will send out a LGA gradient setup a message to override the *global gradient* for the local event report. This is explained in the following paragraph.
- If $R_i < \text{Max}[R_j]$ (j $\in N_i$), then the sensor knows it is not the L_m. It then rebroadcasts the event reports, which have smaller readings with $R_j < R_i$. In this way, the neighbors with $R_j > R_i$ continue to forward these reports until they reach the local maxima.
- If the sensor receives an event report, but it does not have its own reading, it knows it is at the border of LGA and discards the report.

The local maxima sends out Local Gradient Setup (LGS) messages to setup

local gradients, which in the LGA overrides the global gradient. The propagation of LGS messages follows the same rules described in Section 5.3.2. Once the local gradient is established, all the local event reports from LGA flow toward the local maxima, the data aggregation point. Data aggregation performs in-network fusion of data packets, coming from different sensors enroute to the base station, in an attempt to minimize the number and size of data transmissions and thus save sensor energies. Such aggregation can be performed when the data from different sensors are highly correlated as in the LGA. We make the simple assumption that an intermediate sensor can aggregate multiple incoming packets into a smaller number of packets.

Local Maxima Handover

When an event appears in the wireless sensor network, it does not always stay in the same location. Depending on its property, the monitored event could also move in the network, such as in the scenario of intrusion detection, fire control and animal monitoring. This event mobility brings forward a challenge to the efficiency of a routing protocol. In a data aggregation scheme, the efficiency of the aggregation depends highly on the correlation between the gathered data. In our *local gradient*, this means that the closer the local maxima to the center of the event, the higher the correlation of the data. Thus, we designed the *local maxima handover* to allow the local maxima to move together with the center of the event.

Initially the local maxima is the center of the monitored event and it has the largest sensor reading. At the data gathering point, it knows all the readings of the other sensors in the LGA. When the monitored event starts to move, its center gets away from the local maxima. From the readings, the local maxima will notice that it no longer has the largest reading in the LGA and thus is no longer the center of this event. When the differences exceeds a threshold, the local maxima will issue a handover message to the sensor node that has the largest current reading and this sensor will send out a *local gradient* setup message to replace the old local maxima.

Reverse Query Route

When the sink receives aggregated data from the network by the local maxima, it needs a reverse query route to address to the event area. Either additional queries or reinforcements can be sent to the network. Rather than flooding the query update to the whole network, we send it only to the area where the event exists, i.e. the local gradient area. We first forward it to the local maxima and then the local maxima distributes the query update to individual sensors in the local gradient area.

When the intermediate sensors forward aggregated data from the local max-

ima, they keep the state information about this particular event. The state information contains the ID of the local maxima and the type of event. The intermediate sensor updates this state information whenever it forwards data from the same local maxima. The state information expires when there is no forwarding in a defined period. When the sink sends out a query update, only the sensors which have the state information about this event area will forward it. Other sensors ignore the query update. Thus, the query update sent by the sink only travels through the sensors "belt" from the sink to the local maxima via the multiple pathes. Once the local maxima receives a copy of the query update, it disseminates the update to all the sensors in the local event area. When the query update reaches the border of the local event area, it stops. Because the sensors at the border are without local gradient and will not forward the query update.

5.4 Simulation Results

5.4.1 Simulation Testbed

We compared the protocols presented in the previous sections with the GRAdient Broadcast (GRAB) [26] protocol. The same network setup is used to compare the two implementations of routing protocols.

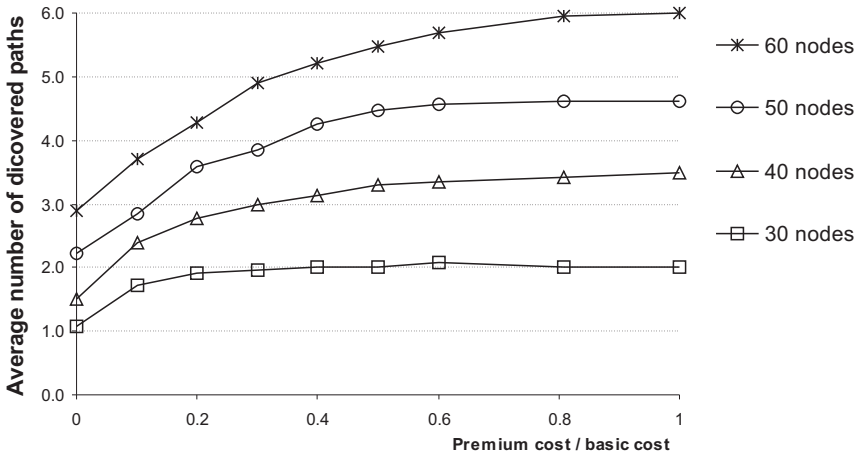


Figure 5.4: The number of multipaths under different premium costs of the data

The OMNeT++ discrete event simulator, together with a framework for a

mobile and wireless network [86], is used in the simulation. In this simulator, a physical layer with energy model is implemented to record the sending and receiving energy consumption of the transceiver. Additionally, switching between sending and receiving takes time and consumes energy, which are also considered in the simulation. The respective data for the transceiver are taken from an RFM TR 1001, which is also used in our prototype sensor nodes (see Table 4.3 on page 99). For each packet, a link failure probability of 0.02 is applied to the physical layer. For both simulations Sensor-MAC protocol (SMAC) [113], a medium access protocol for wireless sensor networks is implemented to provide MAC layer access. It is a Carrier Sense Multiple Access with collision detection (CSMA/cd) protocol. A network of a certain number of sensors with a fixed radio range are randomly placed on a rectangular area. The density of the resulting network is controlled by scaling the area of the model. Reducing the area for node placement yields a denser network, i.e. more neighbors within transmission range. The nodes move in this area according to the random way-point model (RWP) with random speeds and waiting times.

5.4.2 Static network

Premium cost and the multipath degree

In the global data forwarding, the data sent by the source has a cost of its own, which consists of *basic cost* and *premium cost*. The *basic cost* C_{basic}=C_i is the cost of the node that sends the data. It is the minimum cost the data should have in order to reach the sink from node i. The *premium cost* is set to control the multipath degree of the forwarding paths. When the data has more premium cost to spend, it will travel further away from the low cost paths and go through higher cost areas. The premium cost will increase the reliability of the forwarding at the expense of more energy consumption. Thus, the overall cost of the data is expressed as $C_{data} = C_{basic} + C_{premium}$. In this simulation, we try to find the relationship between the *premium cost* and the multipath degree. A network of 30 to 60 sensors with a radio range of 150m are randomly placed on a rectangular area of 800m×800m. We set up only global gradient in the network and allow one random sensor in the network to generate data reports to the data sink. In different simulations, we increased the premium cost from 0 to 100% of the basic cost and compared the number of multipaths in the data forwarding.

As shown in Figure 5.4 on page 135, the number of multipaths discovered by the data packet increases as the amount of premium cost increases. Thus, we could increase the probability of successful delivery of a data by increasing its premium cost. In a dense network, the average multipath degree is larger than in a sparse network. Moreover, there is a upper bound of the multipath degree for a defined

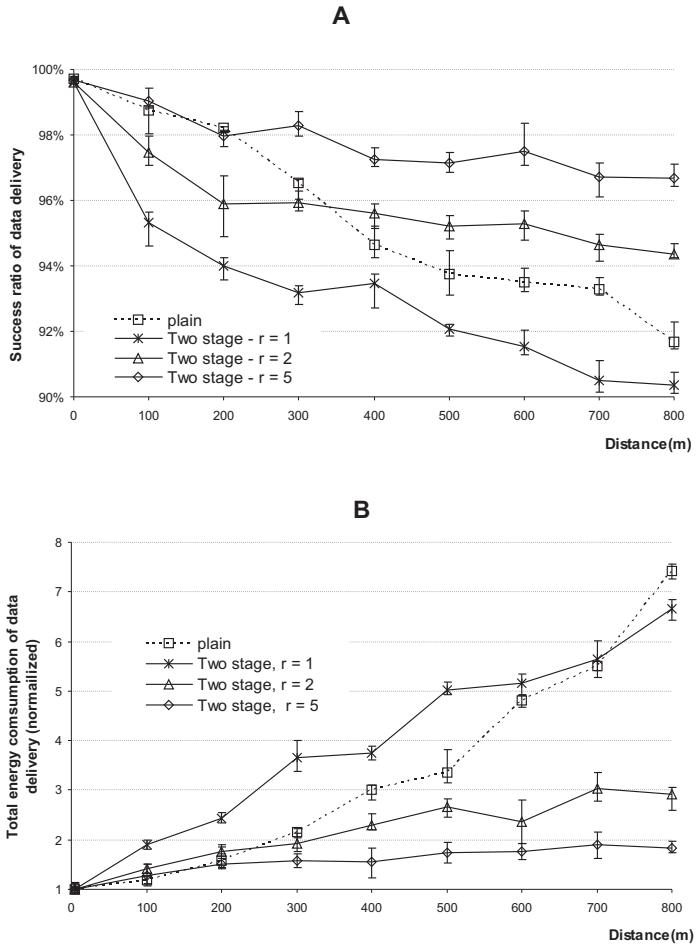


Figure 5.5: The success ratio of data delivery and energy consumption in a static network

network, no matter how large the premium cost is. The maximum number of node disjoint paths is determined by the maximum number of neighbors a data sink has.

Link cost

The metric of link cost decides the forwarding character of the sensor network. In these simulations, we try to show the effect of three different metrics of the link cost: hop count, link quality and energy.

Initially, all the sensor nodes have the same energy budget of E. Each link was assigned a fixed failure probability P between 0 and 0.1. For the three metrics

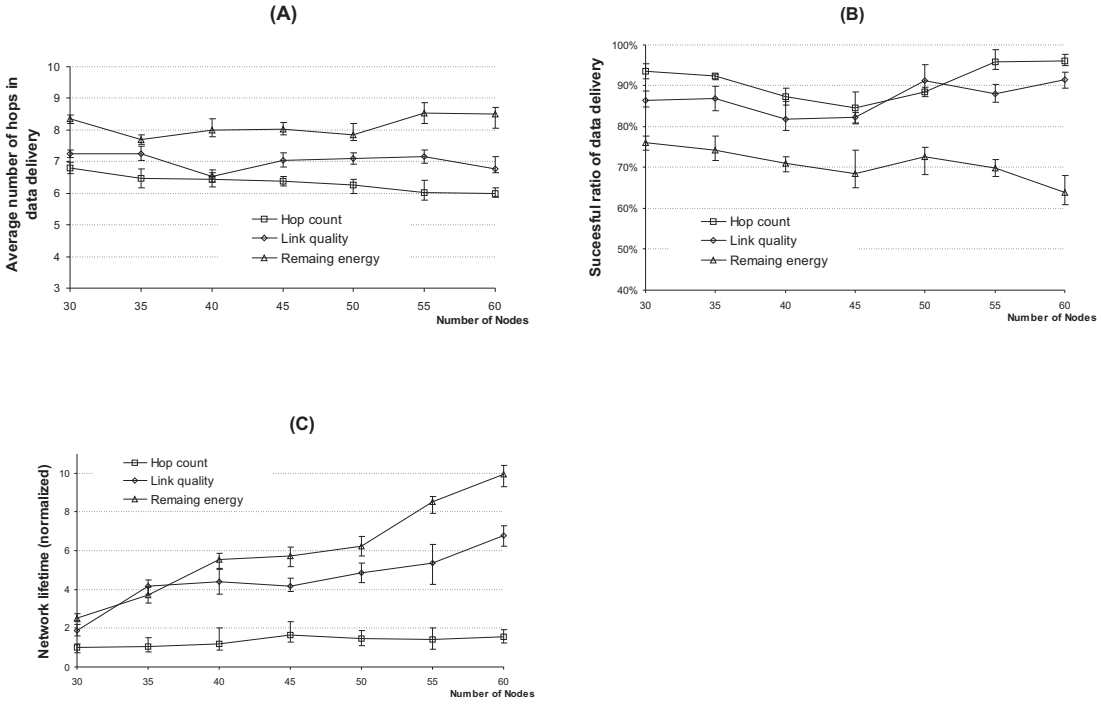


Figure 5.6: The average hop counts , success ratio and network life under different metrics of link cost

of link cost, we ran simulations under the same topology. We choose three simple relationships between the link cost and the metrics to show their effect on the forwarding. For the first simulation, each link had the same link cost $C_{link} = 1$. For the second simulation, the link cost was expressed as $C_{link} = 1+10\text{P}$. For the third simulation, the link cost was expressed as $C_{link} = 2 - E_{remaining}/E$. In Figure 5.6 on page 138, the graph shows the average number of hops in data delivery, the success ratio of data delivery and the network life in the simulations.

The results clearly show that hop count based link cost gives the network a lower delay than the other two metrics. Another observation is that both hop count and reliability based network have a high success ratio of data delivery. The third graph shows that this is achieved through the expense of more energy consumption. The remaining energy based network has the longest network lifetime and it grows with the size of the network.

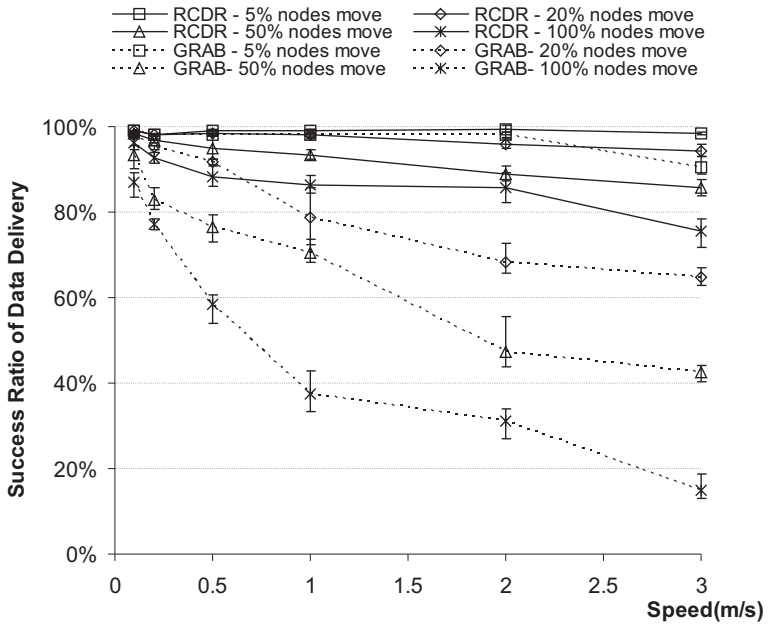


Figure 5.7: The success ratio of of RCDR and GRAB under different speed of sensor movement

Global and local gradients

In our algorithm, the global-local paradigm enables data aggradation close to the event. The idea is to aggregate the data in the local event area and then the local maxima can send the aggregated data through the global gradient to the data sink. In these simulations, we try to find the the reliability of our two-gradient structure and evaluate its energy cost. A network of 60 static sensors with a radio range of 150m are randomly placed on a rectangular area of 600m×1000m. Two sensors are chosen to be the local maxima and the data sink. The distance between them is fixed. The event triggers all the sensors within a fixed radius of 200m and the sensors in the local event area generate a data flow for 10 minutes with a data rate of 1 packet/s. The aggregation ratio of incoming packets over outgoing packets at local maxima is r.

Figure 5.5, on page 137, shows the success ratio of data delivery and energy consumption in the static network. In a plain network, all the sensors send their reports directly to the data sink by the global gradient. From our results, we can clearly see that when there is non aggregation at the local maxima point, our two-stage

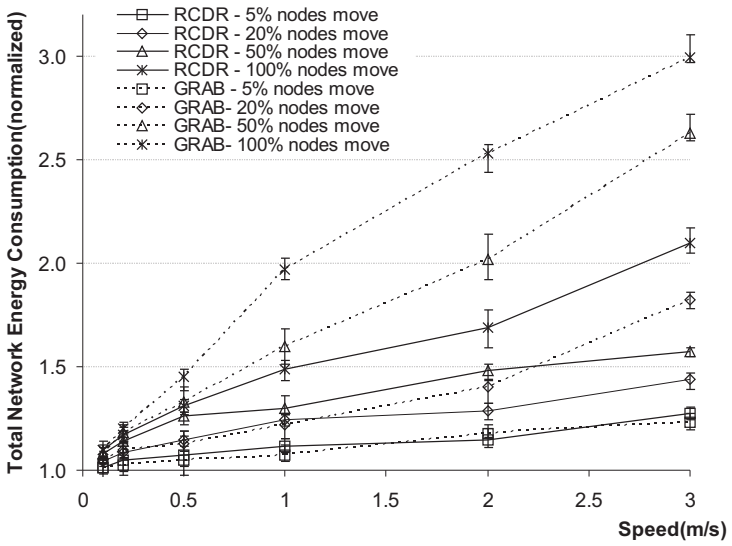


Figure 5.8: The network energy consumption of RCDR and GRAB under different speeds of sensor movement

scheme performs worse than the plain routing. However, when there is aggregation, the reliability increases with the aggregation ratio. At the same time, the energy consumption is much less than in the plain routing. Another important observation is that the global-local gradient paradigm performs well when the distance between the data sink and the event increases. This also indicates that when the network size increases, our scheme has better scalability than plain routing. However, when the data sink is near or inside the local event area, the reliability and energy consumption are worse than with a plain routing. An improvement would be to let the data sink be the local maxima when it notices that it is inside the local event area.

5.4.3 Dynamic network

Sensor movement adjustment scheme

In this simulation, we try to discover the reliability of a *Sensor movement adjustment scheme* under different mobility conditions. A network of 60 sensors with a radio range of 150m is randomly placed on a rectangular area of 800m×800m. We only set up a global gradient in the network and let one random sensor in the network generate data reports to the data sink. This gives a data flow for 10 minutes with

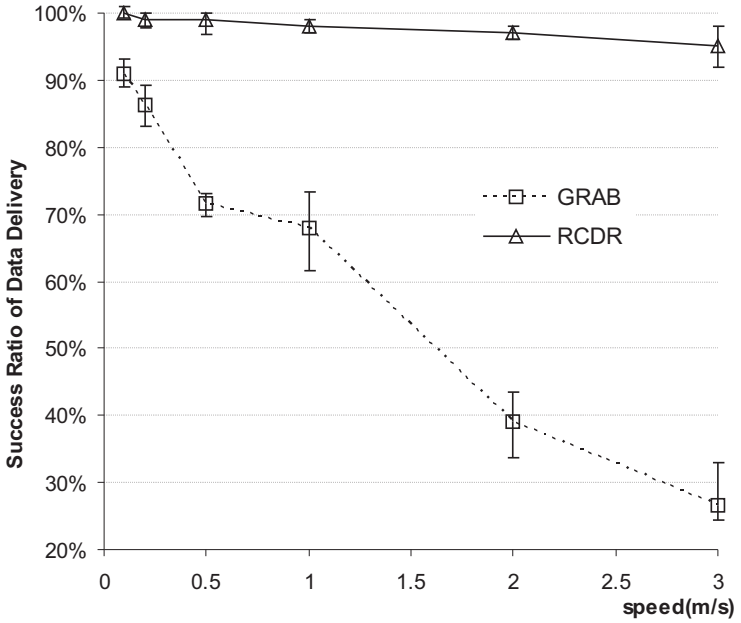


Figure 5.9: The success ratio of RCDR and GRAB under different speeds of sink movement

a data rate of 2 packets/s, after which another random node takes over. A certain percentage of sensors in the network follow the random walk model and move in the network and the other sensors remain static during the course of the simulation.

Firstly, we compared the success ratio of data delivery between RCRD and GRAB under different moving speeds. As shown in Figure 5.7 on page 139, when only 5% of the sensors in the network move with a speed of less than 2m/s, both RCRD and GRAB are rather reliable. When more sensors move in the network, the success delivery by GRAB decreases sharply. And when the speed is more than 2m/s, its success delivery ratio is less than 50%. This means that GRAB is very unreliable and almost non-operational in a dynamic network. On the contrary, RCDR shows a much better resilience to the changing topology due to the *sensor movement adjustment scheme*.

Secondly, we compared the energy consumption of the whole network under different speeds. The simulation time was controlled and the total energy consumption of all the sensors was calculated and normalized against the static GRAB network. As shown in Figure 5.8 on page 140, at low speeds both protocols have similar energy consumption. When the speed increases, GRAB consumes more than two times the amount of energy than RCDR, which is caused by its more frequently

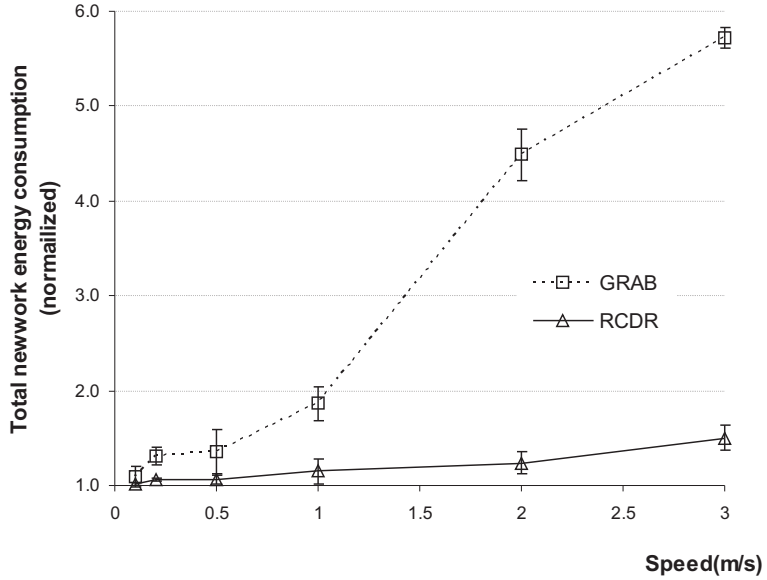


Figure 5.10: The network energy consumption of RCDR and GRAB under different speeds of sink movement

network wide flooding to resume the gradient. However the local adjustment of RCDR shows its advantage at higher speeds in that it saves 50% more on energy consumption.

Sink movement compensation scheme

In this simulation, we tried to find out the reliability of the network under *sink movement compensation scheme*. A network of 60 sensors with a radio range of 150m were randomly placed on a rectangular area of 800m×800m. Again, we only setup a global gradient in the network and allow one random sensor in the network to generate data reports to the data sink. All the sensors, except the data sink, remained static during the course of the simulation. The sink follows a random walk point model with different speeds. Figure 5.9 on page 141 and Figure 5.10 on page 142 clearly show that compared with GRAB, the *sink movement compensation scheme* improves the reliability of the network by 20% at lower speeds and more than 75% at higher speeds. In addition, its energy consumption is only 25%-50% of GRAB. The "disasters" situation of sink movement in GRAB is solved well by our scheme.

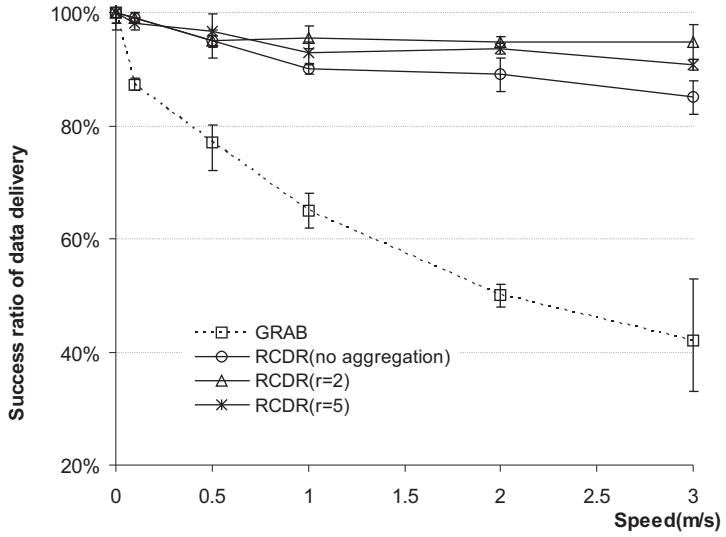


Figure 5.11: The success ratio of RCDR and GRAB under different speeds in a mobile network

Data Aggregation

We make the simple assumption that local maxima sensors can aggregate multiple incoming packets into a fewer number of outgoing packet. The aggregation ratio of incoming packets over outgoing packets r depends on the correlation between the data. A range of [1,5] is selected in the simulation for RCRD. A network of 60 sensors with a radio range of 150m were randomly placed on a rectangular area of 800m×800m. The network randomly produced monitored events of a radius of 200m, which lasted for a random number of minutes between [2, 5]. Only 40% of the sensors in the network moved according to the random walk point model. As any sink movement degraded GRAB significantly, we allowed the sink to remain static all times.

Figure 5.11 on page 143 shows that our protocol has a higher reliability than GRAB under different speeds. Furthermore, the data aggregation ratio has no significant impact on reliability as the aggregated data are protected against error and lost by multipath routing. However, with respect to energy consumption, a higher degree of aggregations clearly gains more from our global-local gradient paradigm, as shown in Figure 5.12 on page 144.

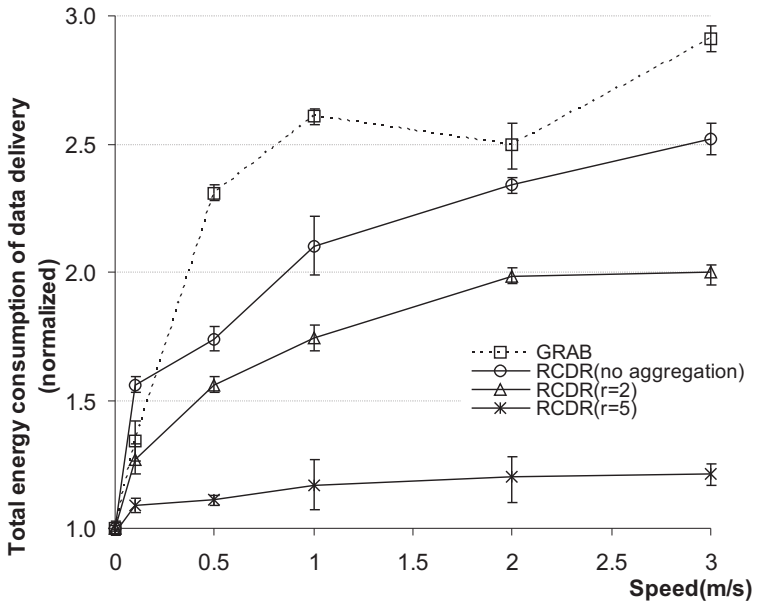


Figure 5.12: The network energy consumption of RCDR and GRAB under different speed in mobile network

5.5 Conclusion

In a dynamic wireless sensor network, the mobility of sensors and the monitored events pose a threat to the efficiency of the routing protocol. In this chapter, we discussed a cost-based data-centric routing protocol, which is a simple yet effective algorithm for nodes to recover a disrupted gradient. Thus, a network-wide flooding is avoided in the maintenance of the gradient field. Again, our solution is only based upon local information. The routing protocol benefits from local topology information that is already present in the medium access protocol. The global-local gradient paradigm ensures that sensors aggregate the collected data as close as possible to its origin, while only a small number of aggregated data are sent to the data sink via multiple reliable and adjustable routes.

The presented approach is compared with GRAdient Broadcast (GRAB) in addition to the SMAC medium access protocol. Simulation results show that our approach to data-centric routing protocol for WSNs benefits clearly from the global-local paradigm we are able to use. In a static network, our approach increases the probability of successful data delivery at least 25%. In a dynamic network topology, the local recovery approaches of our protocol clearly outperform GRAB in reliability while also reducing the energy consumption by 50%.

Chapter 6

Conclusion

A wireless sensor network is a multi-hop ad hoc network of hundreds or thousands of sensor devices. The sensor nodes collect useful information such as sound, temperature, and light. Moreover, they play a role as the router by communicating through wireless channels under battery-constraints. Wireless sensor networks enable the reliable monitoring of a variety of environments for both civil and military applications. In wireless sensor networks, the routing protocol refers to selecting paths in the network along which data is transmitted. Routing directs forwarding, the passing of packets from their source node toward their ultimate destination node through intermediary sensor nodes. In this thesis, we look at routing protocols, which can have a significant impact on the overall reliability and energy dissipation of these networks.

6.1 Evaluation

Taking into account the characteristics of the wireless sensor networks, this thesis addressers the dynamics of sink node, sensor node and event in the routing of wireless sensor network while maintaining high reliability and keeping the energy consumption low. Our hypothesis is that this task requires different routing protocols and approaches. In addition, the different application scenarios of wireless sensor networks require different routing protocols and approaches.

First, we introduced a splitted multipath scheme to control the trade-off between traffic and reliability of data routing in wireless sensor networks. An on-demand multipath routing algorithm offers the data source with several paths to any destination, if available. It is used in combination with a data splitting method based on Modified Erasure Coding. By splitting the data across multiple paths, the

number of errors increases. At the same time, the traffic drops to much lower values when compared with sending the same data across multiple paths. This gives us a way to adjust the reliability while keeping the data traffic low.

We have implemented this scheme and estimated the main characteristics. It greatly increases the reliability of packet delivery in wireless sensor networks, while keeping the total network traffic much lower than in the traditional multipath routing. At the same time, the latency of splitted multipath routing is lower than any retransmission scheme.

In this thesis, we also showed that cross-layer optimization is indeed a useful approach for WSNs. This approach addresses a self-organizing MAC protocol and a tightly integrated, energy efficient routing protocol. It also uses an algorithm to decide the grade of participation of a sensor node to create a connected network based upon local information only. We compared our cross-layer optimized networking protocols with traditional protocols for WSNs: SMAC and DSR. One of the key issues in WSNs is the network lifetime. Simulations show that in equal network configurations, message frequency and size assumptions, our cross-layered approach shows a larger network lifetime, especially in the case when nodes are mobile.

We also implemented the proposed protocol combination in a real-life testbed. The results show that our protocols achieve improved performance on route acquisition time, end-to-end data throughput and delay when the density or the size of the network increases. Moreover, the performance gains become more significant when the topological rate of change is relatively high. Consequently, ESR can dramatically reduce energy consumption and handle the dynamic topology of WSNs in real-world environments.

Finally, in this thesis we discussed a cost-based, data-centric routing protocol, which is a simple yet effective algorithm for nodes to recover the disrupted gradient. In a dynamic wireless sensor network, the mobility of sensors and the monitored events pose a threat to the efficiency of the routing protocol. Thus, a network-wide flooding is avoided in the maintenance of the gradient field. Again, these schemes are only based upon local information. The routing protocol benefits from local topology information that is already present in the medium access protocol. The global-local gradient paradigm ensures that sensors aggregate the collected data as close as possible to its origin, while only a small number of aggregated data are sent to the data sink via multiple reliable and adjustable routes.

The presented approach is compared with GRAdient Broadcast (GRAB) in addition to the SMAC medium access protocol. Simulation results show that our approach to data-centric routing protocol for WSNs benefits clearly from the global-local paradigm we are able to use. While in a static network, our approach increases the network reliability by at least 25%. In a dynamic network topology, the local

recovery approaches of our protocol clearly outrun GRAB in reliability, while also reducing energy consumption to less than 50%.

6.2 Conclusion

The contributions of this thesis are in the dynamic aspect of the wireless sensor network. Although there is some related work in this area, most of it has been based on the geographical information. However, considering the tiny, cheap and resource limited sensor node, additional hardware, such as GPS, is relatively expensive and energy intensive. In this thesis, we contribute research on routing protocols for dynamic wireless sensor networks without geographical information support. As a result of the work in this thesis, three routing approaches have been designed for dynamic wireless sensor networks.

Firstly, our splitted multipath routing scheme achieves the proposed trade-off between traffic and reliability of data routing in wireless sensor networks. Our lessons learned in this research show that more computations in the node can be performed to improve the energy efficiency of WSN. This scheme improves the reliability of dynamic wireless sensor networks in the point-point routing scenario by using multipath routing. This method is suitable to disseminating a large amount of bulk data to the destination with a high reliability and low delay. Our research also showed that a combination of multipath routing with a data-splitting scheme can achieve high delivery ratios while keeping traffic at a low value.

Secondly, our cross-layer approach achieves the proposed reliability improvement in a dynamic wireless sensor network. Our lessons learned in this research show that in the routing protocols for WSN, optimization is more effective when taking into account the overall system and with the use of all available knowledge, instead of a strict layered approach. Our approach is effective in message delivery in the point-point routing scenarios of wireless sensor networks, especially in the application area of high mobility, such as vehicle tracking and personal tracking. This approach has a much larger network lifetime, compared with traditional protocols for WSNs. The implementation results of the cross-layer approach have validated the performance of our design in real network operation. Moreover, the results show that the performance gains of the cross-layer approach become more significant when the speed of the node in the network is relatively high.

Finally, our cost-based data-centric scheme achieves a lower overhead and higher reliability in the data aggregation in a dynamic wireless sensor network compared with the traditional approach. Our lessons learned in this research show that in the data centric routing for WSN, natural diffusion property could be used for

the dissemination of data in a gradient based scheme. We increase the reliability of dynamic wireless sensor networks in the point-multipoint routing scenarios by a data-centric approach. Our approach maintains high reliability and energy efficiency of dynamic wireless sensor networks in the application area such as event tracking and herd monitoring. It can adapt to the topology changes in the network and aggregate the collected data as close as possible to its origin.

6.3 Future work

Our multipath routing with data splitting scheme provides an abstraction of a better transmission medium from the receiver side. In the future, a retransmission scheme could further handle the remaining error corrections. Future work will focus on the hybrid scheme of data splitting and retransmission similar to [103], which will ensure higher reliability in data dissemination. Moreover, future research will also focus on integrating path estimation in the MDR, so that the failure probabilities of each node can be obtained in the routing process. Although focused on WSNs, our scheme can also be incorporated into any routing scheme to improve reliable packet delivery in the face of a dynamic (wireless) environment where nodes move and connections break.

For the reliable source routing protocol, we intend to implement the proposed protocols in combination with a new sensor node prototype with improved radio. A larger testbed of hundreds of nodes will be built and its performance will be evaluated with respect to scalability.

For future work in the cost-based data-centric routing protocol, the effect of event mobility on data aggregation should be investigated more. Many applications of wireless sensor networks focus on the monitoring of events with high degrees of mobility. A dynamic in-network aggregation scheme should benefit from local data processing. Therefore, a solution is required to improve the reliability and energy efficiency in this direction.

Many different application scenarios for wireless sensor networks are possible. It is unlikely that there will be a one-thing-fits-all solution for all these potentially very different possibilities. Routing protocols should be able to adapt to the application requirements. Even in the same application, the characteristic of the network may dramatically change with time or space, either periodically or randomly. Another direction for future work is to improve the current routing protocols to better adapt to these changes in the wireless sensor network.

Bibliography

[1] Omnet++ discrete event simulation system. http://www.omnetpp.org/.

[2] Rev. A. Mpr/mib user's manual. *Document 7430-0021-06. Crossbow Technology, Inc.*, 2004.

[3] K. Akkaya and M. Younis. A survey of routing protocols in wireless sensor networks. *Ad Hoc Network*, 3(3):325–349, 2005.

[4] I. Akyildiz, W. Su, Y. Sankarasubramaniam, , and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, 2002.

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

[6] A. D. Amis, R. Prakash, D. Huynh, and T. Vuong. Max-min d-cluster formation in wireless ad hoc networks. In *Proceedings of the Ninteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Tel-Aviv, Israel*, volume 1, pages 32–41. IEEE Computer Society, 2000.

[7] S. Basagni. Finding a maximal weighted independent set in wireless networks. *Telecommunication Systems, Special Issue on Mobile Computing and Wireless Networks*, 18(1/3):155–168, 2001.

[8] G. E. Blomgren. Current status of lithium ion and lithium polymer secondary batteries. In *Proceedings of IEEE Fifteenth Annual Battery Conference on Applications and Advances, Long Beach, USA*, pages 16–23. IEEE Computer Society, 2000.

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

[10] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the fourth annual ACM/IEEE international conference on Mobile computing and networking , Dallas, USA*, pages 85–97. ACM Press, 1998.

[11] J. Broch, D.A. Maltz, D.B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the fourth annual ACM/IEEE international conference on Mobile computing and networking, Dallas, USA*, pages 85–97. ACM Press, 1998.

[12] BTnodes - A Distributed Environment for Prototyping Ad Hoc Networks. website. `http://www.btnode.ethz.ch/`.

[13] B. H. Calhoun, D. C. Daly, N. Verma, D. Finchelstein, D. D. Wentzloff, A. Wang, S.-H. Cho, and A. P. Chandrakasan. Design considerations for ultra-low energy wireless microsensor nodes. *IEEE Transactions on Computers*, 54(6):727–740, 2005.

[14] J. Chen, P. Druschel, and D. Subramanian. An efficient multipath forwarding method. In *Proceedings of the Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies, San Francisco, USA*, volume 3, pages 1418–1425. IEEE Computer Society, 1998.

[15] Tsu-Wei Chen and Mario Gerla. Global state routing: A new routing scheme for ad-hoc wireless networks. In *Proceedings of the IEEE International Conference on Communications (ICC), Atlanta, USA*, volume 8, pages 171–175. IEEE Computer Society, 1998.

[16] C.-C. Chiang. Routing in clustered multihop, mobile wireless networks with fading channel. In *Proceedings of IEEE International Conference on Networks, Singapore*, pages 197–211. IEEE Computer Society, 1997.

[17] I. Cidon, R. Rom, and Y. Shavitt. Analysis of multi-path routing. *IEEE/ACM Transactions on Networking*, 7(6):885–896, 1999.

[18] G. C. Clark and J. B. Cain. *Error Correction Coding for Digital Communications*. Perseus Publishing, 1988.

[19] L. Doherty, K. Pister, and L.El Ghaoui. Convex position estimation in wireless sensor networks. In *Proceedings of Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), Anchorage, USA*, volume 3, pages 1655–1663. IEEE Computer Society, 2001.

[20] D. Dubhashi, A. Mei, A. Panconesi, J. Radhakrishnan, and A. Srinivasan. Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons. *Proceedings of the Fourteenth ACM-SIAM Symposium on Discrete Algorithms, Baltimore, USA*, 72(4):467–479, 2003.

[21] S. Dulman. Data-centric architecture for wireless sensor networks. *PhD. Dissertation*, pages 59–94, 2005.

[22] S. Dulman, T. Nieberg, J. Wu, and P. Havinga. Trade-off between traffic overhead and reliability in multipath routing for wireless sensor networks. In *Proceedings of the Wireless Communications and Networking Conference*, volume 3, pages 1918– 1922. IEEE Computer Society, 2003.

[23] P. Lassila A. Penttinen E. Hyyti, H. Koskinen and J. Virtamo. Random waypoint model in wireless networks. In *Proceedings of Networks and Algorithms: complexity in Physics and Computer Science, Helsinki, Finland*, pages 132–156. ACM Press, 2005.

[24] I.A. Essa. Ubiquitous sensing for smart and aware environments. *IEEE Personal Communications*, 7(5):47–49, 2000.

[25] EYES project. website. `http://eyes.eu.org`.

[26] Songwu Lu Fan Ye, Gary Zhong and Lixia Zhang. A robust data deliver protocol for large scale sensor networks. In *Proceedings of the second International Workshop on Information Processing in Sensor Networks (IPSN), Palo Alto, USA*, 2003.

[27] J. Faruque and A. Helmy. Rugged: Routing on fingerprint gradients in sensor networks. In *Proceedings of IEEE International Conference on Pervasive Services (ICPS), Lebanon*, pages 179– 188. IEEE Computer Society, 2004.

[28] Krisztin Flautner, Steve Reinhardt, and Trevor Mudge. Automatic performance setting for dynamic voltage scaling. *Wireless Networks*, 8(5):507–520, 2002.

[29] Deepak Ganesan, Ramesh Govindan, Scott Shenker, and Deborah Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(4):11–25, 2001.

[30] M. Gerla and T.-J. Tsai. Multicluster, mobile, multimedia radio network. *ACM/Baltzer Journal of Wireless Networks*, 1(3):255–265, 1995.

[31] John Heidemann, Fabio Silva, Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, and Deepak Ganesan. Building efficient wireless sensor networks with low-level naming. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles, Alberta, Canada*, pages 146–159. ACM Press, 2001.

[32] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd International Conference on System Sciences (HICSS), Hawaii, USA*, pages 56–73. IEEE Computer Society, 2000.

[33] W. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of the Fifth ACM/IEEE Mobicom Conference (MobiCom), Seattle, USA*, page 174C185. ACM Press, 1999.

[34] W. R. Heizelman, A. Chandrakasan, and H. Balakrishnan. Directed diffusion: A salable and robust communication paradigm for sensor networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking, Boston, USA*, volume 3, pages 56–67. ACM Press, 2000.

[35] W. R. Heizelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless micro-sensor networks. In *Proceedings of the 33rd Annual Hawaii International Conference on, Hawaii, USA*, volume 10, pages 6–10. IEEE Computer Society, 2000.

[36] Thomas C. Henderson and Eddie Grant. Gradient calculation in sensor networks. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan*, volume 2, pages 1792–1795. IEEE Computer Society, 2004.

[37] Radhakrishna Hiremane. *Technology@Intel Magazine*, 2005. `http://www.intel.com/technology/magazine/silicon/moores-law-0405.pdf`.

[38] G. Hoblos, M. Staroswiecki, and A. Aitouche. Optimal design of fault tolerant sensor networks. In *Proceedings of IEEE International Conference on Control Applications, Anchorage, AK*, pages 467–472. IEEE Computer Society, 2000.

[39] L.V. Hoesel, T. Nieberg, J. Wu, and P. Havinga. Prolonging the lifetime of wireless sensor networks by cross-layer interaction. *the Special Issue on Wireless Sensor Networks, IEEE Wireless Communication Magazine*, 11(6):78–86, 2004.

[40] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking, Boston, USA*, volume 3, pages 56–67. ACM Press, 2000.

[41] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networks*, volume 3, pages 56–67. ACM Press, 2000.

[42] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark. Scenario-based performance analysis of routing protocols for mobile ad-hoc networks. In *Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking, Seattle, USA*, pages 195 – 206. ACM Press, 1999.

[43] D. Johnson, Y. Hu, and D. Maltz. The dynamic source routing protocol for mobile ad hoc networks. *IETF Internet draft*, 2003.

[44] David B. Johnson. Routing in ad hoc networks of mobile hosts. In *Proceedings of the Workshop on Mobile Computing Systems and Applications, USA*, pages 158–163. IEEE Computer Society, 1994.

[45] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 53-181. Kluwer Academic Publishers, 1996.

[46] C. Kappler and G. Riegel. A real-world, simple wireless sensor network for monitoring electrical energy consumption. In *Proceedings of the first European Workshop on Wireless Sensor Networks, Berlin, Germany*, volume 2920/2004, pages 339–352. Springer Berlin / Heidelberg, 2004.

[47] S. Kininmonth, S. Bainbridge, I. Atkinson, E. Gill, L. Barral, and R. Vidaud. Sensor networking the great barrier reef. *Spatial Sciences Institute Journal (QLD)*, 50(2):101–129, 2005.

[48] R. Kling. Intel mote: An enhanced sensor network node. In *Proceedings of International Workshop on Advanced Sensors, Structural Health Monitoring and Smart Structures, Stanford, USA*, 2003.

[49] B. Krishnamachari, D. Estrin, and S. Wicker. Modelling data-centric routing in wireless sensor networks. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies, New York, USA*, volume 2, pages 39– 44. IEEE Computer Society, 2002.

[50] Bhaskar Krishnamachari, Deborah Estrin, and Stephen B. Wicker. The impact of data aggregation in wireless sensor networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems, USA*, pages 575–578. IEEE Computer Society, 2002.

[51] F. Kuhn, R. Wattenhofer, and A. Zollinger. Worst-case optimal and average-case efficient geometric ad-hoc routing. In *Proceedings of Fourth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC), Annapolis, USA*, pages 267–278. ACM Press, 2003.

[52] J. Kulik, W. R. Heinzelman, and H. Balakrishnan. Negotiation-based protocols for disseminating information in wireless sensor networks. *Wireless Networks Magazine*, 8:169–185, 2002.

[53] S.-J. Lee and M. Gerla. Split multipath routing with maximally disjoint paths in ad hoc networks. In *Proceedings of IEEE International Conference on Communications, Helsinki, Finland*, volume 10, pages 3201–3205. IEEE Computer Society, 2001.

[54] S. Lindsey and C. S. Raghavendra. Pegasis: Power efficient gathering in sensor information systems. In *Proceedings of the IEEE Aerospace Conference, Big Sky, USA*, volume 3, pages 1125–1130. IEEE Computer Society, 2002.

[55] S. Lindsey, C. S. Raghavendra, and K. Sivalingam. Data gathering in sensor networks using the energy-delay metric. *Proceedings of the IPDPS Workshop on Issues in Wireless Networks and Mobile Computing, San Francisco, USA*, 13(9):924–935, 2001.

[56] K. Lorincz and M. Welsh. Motetrack: A robust, decentralized approach to rf-based location tracking. In *Proceedings of the International Workshop on Location and Context-Awareness (LoCA 2005) at Pervasive 2005, USA*, volume 3479 of *Lecture Notes in Computer Science*, pages 63–82. Springer, 2005.

[57] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, , and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the ACM International Workshop on Wireless Sensor Networks and Applications, Atlanta, USA*, pages 88–97. ACM Press, 2002.

[58] A. Manjeshwar and D. P. Agarwal. Teen: a routing protocol for enhanced efficiency in wireless sensor networks. In *Proceedings of 1st International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing, San Francisco, USA*, pages 2009–2015. IEEE Computer Society, 2001.

[59] A. Manjeshwar and D. P. Agarwal. Apteen: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks. In *Proceedings of International Parallel and Distributed Processing Symposium (IPDPS), Fort Lauderdale, USA*, pages 195–202. IEEE Computer Society, 2002.

[60] M. Marin-Perianu and P. J. M.Havinga. Experiments with reliable data delivery in wireless sensor networks. In *Proceedings of IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), Melbourne, Australia*, pages 109– 114. IEEE Computer Society, 2005.

[61] M.K. Marina and S.R. Das. On-demand multipath distance vector routing in ad hoc networks. In *Proceedings of the Ninth International Conference on Network Protocols, Washington, USA*, page 14. IEEE Computer Society, 2001.

[62] K. Martinez, R. Ong, J. K. Hart, and J. Stefanov. Glacsweb: A sensor web for glaciers. In *Proceedings of the first European Workshop on Wireless Sensor Networks, Berlin, Germany*, volume 2920 of *Lecture Notes in Computer Science*, pages 234–237. Springer, 2004.

[63] W. M. Meriall, F. Newberg, K. Sohrabi, W. Kaiser, , and G. Pottie. Collaborative networking requirements for unattended ground sensor systems. *Proceedings of IEEE Aerospace Conference, Montana, Canada*, 5(2153-2165), March, 2003.

[64] F. Michahelles, P. Matter, A. Schmidt, and B. Schiele. Applying wearable sensors to avalanche rescue. *Computers and Graphics*, 27(6):839–847(9), December 2003.

[65] Moteiv Corporation. website. `http://www.moteiv.com/`.

[66] J. Moy. Ospf version 2. *RFC 1247*, 1991.

[67] S. Murthy and J.J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *ACM Mobile Networks and App. J., Special Issue on Routing in Mobile Communication Networks*, 1(2):183–197, 1996.

[68] Young Han Nam, Zeehun Halm, Young Joon Chee, and Kwang Suk Park. Development of remote diagnosis system integrating digital telemetry for medicine. *Proceedings of the 20th Annual International Conference of the IEEE, Hong Kong*, 3(4):1170–1173, 1998.

[69] Nasipuri and S. Das. On-demand multipath routing for mobile ad hoc networks. In *Proceedings of the Eighth International Conference on Computer Communications and Networks (IC3N 99)*, pages 64–70. IEEE Computer Society, 1999.

[70] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proceedings of the fifth annual International Conference on Mobile Computing and Networking, Seattle, USA*, pages 151–162. ACM Press, 1999.

[71] T. Nieberg. Independent and dominating sets in wireless communication graphs. *PhD. Dissertation*, pages 91–113, 2006.

[72] S. Nikolaidis and T. Laopoulos. Instruction-level power consumption estimation of embedded processors for low-power applications. *Journal on Computer Standards and Interfaces*, 24(2):133–137, 2002.

[73] V. D. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proceedings of the Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1405–1413. IEEE Computer Society, 1997.

[74] Charles E. Perkings, Elizabeth M.Royer, and Samir R.Das. Performance comparison of two on-demand routing protocols for ad hoc network. *IEEE Personal Communications*, 1:16–28, 2001.

[75] C. Perkins. *Ad Hoc Networks.* Addison-Wesley, 2000.

[76] C. Perkins, E. Royer, and S. Das. Ad hoc on demand distance vector (aodv) routing. *IETF Internet draft*, 2003.

[77] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. In *Proceedings of the ACM Special Interest Group on Data Communications (SIGCOMM)*, pages 234–244. ACM Press, 1994.

[78] Charles E. Perkins. Ad hoc on demand distance vector (aodv) routing. *IETF Internet draft*, 1997.

[79] Joseph Polastre, Robert Szewczyk, Cory Sharp, and David Culler. The mote revolution: Low power wireless sensor network devices. In *Proceedings of Hot Chips 16: A Symposium on High Performance Chips, Stanford, USA*, pages 56–76. IEEE Computer Society, 2004.

[80] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, 2000.

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

[82] V. Raghunathan, C. Schurgers, Park.S, and M.B. Srivastava. Energy-aware wireless microsensor networks. *IEEE Signal Processing Magazine*, 19(2):40–50, 2002.

[83] K. Romer. The lighthouse location system for smart dust. In *Proceedings of ACM/USENIX Conference on Mobile Systems, Applications and Services (MobiSys), San Francisco, USA*, pages 15–30. ACM Press, 2003.

[84] C. Schurgers and M.B. Srivastava. Energy efficient routing in wireless sensor networks. In *Proceedings of MILCOM Communications for Network-Centric Operations: Creating the Information Force, Vienna, Austria*, pages 357–361, 2001.

[85] S.Dulman and P.Havinga. A simulation template for wireless sensor networks. In *Proceedings of the Supplement of the The Sixth International Symposium on Autonomous Decentralized Systems, Pisa, Italy.* IEEE Computer Society, 2003.

[86] S.Dulman and P.Havinga. A simulation template for wireless sensor networks. In *Proceedings of the Supplement of the The Sixth International Symposium on Autonomous Decentralized Systems, Pisa, Italy.* IEEE Computer Society, 2003.

[87] Li Shang, Li-Shiuan Peh, and Niraj K. Jha. Dynamic voltage scaling with links for power optimization of interconnection networks. In *Proceedings of the 9th International Symposium on High-Performance Computer Architecture, Washington,USA*, pages 91– 102. IEEE Computer Society, 2003.

[88] Y. Shang, W. Ruml, Y. Zhang, and M.P.J. Fromherz. Localization from mere connectivity. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc), Annapolis, USA*, pages 201–212. ACM Press, 2003.

[89] Eugene Shih, SeongHwan Cho, Nathan Ickes, Rex Min, Amit Sinha, Alice Wang, and Anantha Chandrakasan. Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks. In *Proceedings of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom), Rome, Italy*, pages 272 – 287. ACM Press, 2001.

[90] Gyula Simon, Miklos Maroti, Akos Ledeczi, Gyorgy Balogh, Branislav Kusy, Andras Nadas, Gabor Pap, Janos Sallai, and Ken Frampton. Sensor network-based countersniper system. In *Proceedings of the 2nd international conference on Embedded networked sensor systems, 2004, Baltimore, USA*, pages 1–12. ACM Press, 2004.

[91] Paul Havinga Stefan Dulman, Tjerk Hofmeijer. Ambientrt - real time, data centric system software for wireless sensor networks. In *Proceedings of the 21st sensor symposium on sensors, micromachine and applied systems (SMAS)*, 2004.

[92] I. Stojmenovic and X. Lin. Gedir: Loop-free location based routing in wireless networks. In *Proceedings of International Conference on Parallel and Distributed Computing and Systems, Boston, USA*, pages 173–180. IEEE Computer Society, 1999.

[93] N. Taft-Plotkin, B. Bellur, and R. Ogier. Quality-of-service routing using maximally disjoint paths. In *Proceedings of the Seventh International Workshop on Quality of Service, London, UK*, pages 119–128. IEEE Computer Society, 1999.

[94] C. Talarico, J.W. Rozenblit, V. Malhotra, and A. Stritter. A new framework for power estimation of embedded systems. *Computer*, 38(2):71 – 78, 2005.

[95] C.K. Toh. *Ad Hoc Mobile Wireless Networks: Protocols and Systems. Age of Pervasive Mobile Networking and Computing.* Prentice-Hall, 2002.

[96] L. van Hoesel and P. Havinga. Medium access protocol for eyes wireless sensor nodes. In *Proceedings of Tales of the Disappearing Computer, Greece*, 2003.

[97] L. van Hoesel and P. Havinga. A lightweight medium access protocol (lmac) for wireless sensor networks. In *Proceedings of the first International Workshop on Networked Sensing Systems (INSS), Tokyo, Japan*, 2004.

[98] L.F.W. van Hoesel. Senosor on speaking terms - schedule-based medium access protocol for wsn. *PhD. Dissertation*, pages 171–177, 2007.

[99] 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, Centre for Telematics and Information Technology, University of Twente*, 2003.

[100] Andras Varga. The omnet++ discrete event simulation system. In *European Simulation Multiconference (ESM'2001), Prague, Czech Republic*, 2001.

[101] S. Vutukury and J.J. Garcia-Luna-Aceves. An algorithm for multipath computation using distance-vectors with predecessor information. In *Proceedings of the IEEE International Conference on Computer Communications and Networks, Boston, USA*, pages 534–539. IEEE Computer Society, 1999.

[102] A. Wang, S-H. Cho, C. G. Sodini, and A. P. Chandrakasan. Energy efficient modulation and mac for asymmetric microsensor systems. In *Proceedings of The International Symposium on Low Power Electronics and Design (ISLPED), California, USA*, pages 106–111. IEEE Computer Society, 2001.

[103] L. Wang and Sandeep S. Kulkarni. Proactive reliable bulk data dissemination in sensor networks. In *Proceedings of the IASTED Conference on Parallel and Distributed Computing and Systems, Phoenix, USA*, pages 773–778. IASTED/ACTA Press, 2005.

[104] M. Weiser. The computer for the twenty-first century. *Scientific American*, 265(3):94–10, 1991.

[105] M. Weiser. Hot topics: Ubiquitous computing. *Computer*, 26(10):71–72, 1993.

[106] A. Woo and D. Culler. A transmission control scheme for media access in sensor networks. In *Proceedings of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom), Rome, Italy*, pages 221–235. ACM Press, 2001.

[107] Jian Wu, Stefan Dulman, and Paul Havinga. Reliable splitted multipath routing for wireless sensor network. In *Proceedings of IFIP International Conference on Network and Parallel Computing, Wuhan, China*, Lecture Notes in Computer Science, pages 592–600. Springer-Verlag GmbH, 2004.

[108] Jian Wu and Paul Havinga. Reliable cost-based data-centric routing protocol for wireless sensor networks. In *Proceedings of 7th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, Las Vegas, USA*, pages 267 – 272. IEEE Computer Society, 2006.

[109] Jian Wu, Paul Havinga, Stefan Dulman, and Tim Nieberg. Eyes source routing protocol for wireless sensor networks. In *Proceedings of the First European Workshop on Wireless Sensor Networks, Berlin, Germany*, volume 2920 of *Lecture Notes in Computer Science*, pages 45–51. Springer, 2004.

[110] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad-hoc routing. In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking, Rome, Italy*, pages 70–84. ACM Press, 2001.

[111] F. Ye, S. Lu, and L. Zhang. Gradient broadcast: A robust,long-live large sensor network. *Wireless Networks*, 11(3):285 – 298, 2001.

[112] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang. A two-tier data dissemination model for large-scale wireless sensor networks. In *Proceedings of the Eighth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM), Atlanta, USA*, pages 148–159. ACM Press, 2002.

[113] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, New York, USA*, volume 3, pages 1567–1576. IEEE Computer Society, 2002.

[114] Z. Ye, S.V. Krishnamurthy, and S.K. Tripathi. A framework for reliable routing in mobile ad hoc networks. In *Proceedings of Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies, San Francisco, USA*, volume 1, pages 270– 280. IEEE Computer Society, 2003.

[115] Y. Yu, D. Estrin, and R. Govindan. Geographical and energy-aware routing: A recursive data dissemination protocol for wireless sensor networks. *UCLA Computer Science Department Technical Report, UCLA-CSD TR-01-0023*, 2001.

[116] W.T. Zaumen and J.J. Garcia-Luna-Aceves. Loop-free multipath routing using generalized diffusing computations. In *Proceedings of the Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies, San Francisco, USA*, volume 3, pages 1408–1417. IEEE Computer Society, 1998.

[117] Pei Zhang, Christopher M. Sadler, Stephen A. Lyon, and Margaret Martonosi. Hardware design experiences in zebranet. In *Proceedings of 2nd International Conference on Embedded Networked Sensor Systems, USA*, pages 227–238. ACM Press, 2004.

[118] Michele Zorzi and Ramesh R. Rao. Geographic random forwarding (geraf) for ad hoc and sensor networks: Energy and latency performance. *IEEE Transactions on Mobile Computing*, 2(4):337–348, 2003.

Appendix A

List of Publications

- Reliable Cost-based Data-centric Routing Protocol for Wireless Sensor Networks. With P. Havinga. *In the Proceedings of International Workshop on Self-Assembling Wireless Networks*, Las Vegas, USA, June 2006.

- Implementation of an On-Demand Routing Protocol for Wireless Sensor Networks. With Y. Zhang and P. Havinga, *In the Proceedings of 13th International Conference on Telecommunications*, Portugal, May 2006.

- Prolonging the Lifetime of Wireless Sensor Networks by Cross-layer Interaction. With L.V. Hoesel, T. Nieberg and P. Havinga, *In the Special Issue on Wireless Sensor Networks, IEEE Wireless Communication Magazine*, ISBN 1536-1284, Vol. 11, No. 6, p. 78-86, U.S.A., December 2004.

- Multipath Routing with Erasure Coding for Wireless Sensor Networks. With S. Dulman, P. Havinga and T. Nieberg, *In the Proceedings of ProRISC Workshop 2004*, Veldhoven, The Netherlands, ISBN: 90-73461-42-1, November 2004.

- Reliable Splitted Multipath Routing for Wireless Sensor Networks. With S. Dulman and P. Havinga, *In the Proceedings of IFIP International Conference on Network and Parallel Computing*, Wuhan, China, ISBN 3-540-23388-1, October 2004.

- Communication in the EYES Wireless Sensor Network: Tight Integration of Networking Layers Extends Lifetime. With L.F.W. van Hoesel, T. Nieberg and P.J.M. Havinga, *In the Proceedings of International Workshop on Wireless Ad hoc Networks*, Finland, June 2004

- EYES Source Routing Protocol for Wireless Sensor Networks. With P. Havinga, S. Dulman and T. Nieberg, *In the Proceedings of European Workshop on Wire-*

less Sensor Networks (EWSN), ISBN 3-540-20825-9, Berlin, Germany, January 2004.

- Collaborative Communication Protocols for Wireless Sensor Networks. With T.Nieberg, S.Dulman, P.Havinga and L.van Hoesel, *In the book of Ambient Intelligence: Impact on Embedded Systems*, edited by T.Basten and M.Geilen and H.de Groot, Kluwer Academic Publishers, ISBN1-4020-7668-1, November 2003.

- Energy Efficient Routing in Wireless Sensor Networks. With P. Havinga, S. Dulman and T. Nieberg, *In the Proceedings of ProRISC Workshop 2003*, Veldhoven, The Netherlands, ISBN: 90-73461-39-1, November 2003.

- An Energy Efficient Multipath Routing Algorithm for Wireless Sensor Networks. With S. Dulman and P. Havinga, *In the Proceedings of IEEE International Symposium on Autonomous Decentralized Systems (ISADS)*, Pisa, Italy, ISBN-0-7695-1876-1, April 2003.

- Trade-Off between Traffic Overhead and Reliability in Multipath Routing for Wireless Sensor Networks. With S. Dulman, T. Nieberg, and P. Havinga, I*n the Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, New Orleans, USA, ISBN 0-7803-7701-X, May 2003.

Appendix B

List of Abbreviations

AFR	Adaptive Face Routing
ATM	Asynchronous Transfer Mode
APTEEN	Adaptive Threshold sensitive Energy Efficient sensor Network protocol
AODV	Ad hoc On-demand Distance Vector routing
CGSR	Cluster-head Gateway Switch Routing
DD	Directed diffusion
DSDV	Destination-Sequenced Distance-Vector routing
DSR	Dynamic Source Routing
EMAC	EYES Medium Access Control protocol
ESR	EYES Source Routing
GAF	Geographic Adaptive Fidelity
GBR	Gradient-Based Routing
GEAR	Geographic and Energy Aware Routing
GeRaf	Geographic Random Forwarding
GOAFR	Greedy Other Adaptive Face Routing
GRAB	GRAdient Broadcast
GSR	Global State Routing
LCC	Least Cluster Change
LEACH	Low Energy Adaptive Clustering Hierarchy
LMAC	Lightweight Medium Access Control protocol
MAC	Medium Access Control
MCU	MicroController Unit
MDR	Multipath on-Demand Routing
MEC	Modified Erasure Correction code
OFR	Other Face Routing
OSPF	Open Shortest Path First routing
PEGASIS	Power-Efficient GAthering in Sensor Information Systems

RCDR	Reliable Cost-based Data-centric Routing
RSC	Reed-Solomon Code
SPIN	Sensor Protocols for Information via Negotiation
SMAC	Sensor Medium Access Control protocol
SMR	Split Multipath Routing
TDMA	Time Division Multiple Access
TEEN	Threshold-sensitive Energy Efficient sensor Network protocol
TTDD	Two-Tier Data Dissemination
WRP	Wireless Routing Protocol
WSN	Wireless Sensor Networks