Sleeping Strategies for Wireless Sensor Networks

by

Ou Yang

Submitted in Partial Fulfillment
of the
Requirements for the Degree
Doctor of Philosophy

Supervised by
Professor Wendi B. Heinzelman

Department of Electrical and Computer Engineering
Arts, Sciences and Engineering
Edmund A. Hajim School of Engineering and Applied Sciences

University of Rochester
Rochester, New York

2011

Curriculum Vitae

The author received her Bachelor of Science degree from Beijing University of Posts and Telecommunications in 2003. She attended Tsinghua University, Beijing, China from 2003 to 2006 and graduated with a Master of Science degree in Electronics Engineering in 2006. She began graduate studies at the University of Rochester in 2006 and received the Master of Science degree from the University of Rochester in 2008. She had a research internship with Aspera, Inc. during the summer of 2010.

List of Publications and Articles Submitted for Publication:

Ou Yang, Wendi Heinzelman, "Sleeping Multipath Routing: A Trade-off Between Reliability and Lifetime in Wireless Sensor Networks," accepted for publication in Global Communication Conference (GlobeCom) 2011.

Ou Yang, Wendi Heinzelman, "Modeling and Performance Analysis for Duty-cycled MAC Protocols in Wireless Sensor Networks," accepted for publication in IEEE Transactions on Mobile Computing, 2011.

Ou Yang, Wendi Heinzelman, "Modeling and Throughput Analysis for X-MAC with a Finite Queue Capacity," Global Communication Conference (GlobeCom) 2010.

Ou Yang, Wendi Heinzelman, "Modeling and Throughput Analysis for SMAC with a Finite Queue Capacity," International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP) 2009.

Ou Yang, Wendi Heinzelman, "A Better Choice for Sensor Sleeping," European Conference on Wireless Sensor Networks (EWSN) 2009.

Ou Yang, Wendi Heinzelman, "Sensor Selection Cost Function to Increase Network Lifetime with QoS Support," ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM) 2008.

Ou Yang, Chris Merlin, Wendi Heinzelman, "A General Cost Function to Reflect Sensor Support for Application QoS," poster, International Conference on Distributed Computing in Sensor Systems (DCOSS) 2007.

Ou Yang, Jianhua Lu, "Call Admission Control and Scheduling Schemes with QoS Support for Real-time Video Applications in IEEE 802.16 Networks," Journal of Multimedia, Vol. 1, No. 2, May 2006.

Ou Yang, Jianhua Lu, "A New Scheduling and CAC Scheme for Real-Time Video

Application in Fixed Wireless Networks," IEEE Consumer Communications and Networking Conference (CCNC) 2006.

Acknowledgments

Foremost, I would like to thank my advisor and mentor, Prof. Wendi Heinzelman, for her guidance, encouragement, and continuous support to my Ph.D. study and research. Wendi's motivation, enthusiasm, patience, and immense knowledge make her a terrific advisor. I appreciate her professional skills in doing research and her passionate attitude in living life and caring for family. Wendi sets a role model for me during my studies at the University of Rochester and in my future career.

My sincere thanks also go to the many great faculty members who I worked with over the years: Prof. Azadeh Vosoughi, who served on my thesis and annual review committee, Prof. Alireza Seyedi, who served on my annual review committee, Prof. Sandhya Dwarkadas, who served on my thesis committee, Prof. Henry Kautz, who served as my thesis committee chair, and Prof. Michael Huang, who was a mentor to me as a teaching assistant.

I would like to thank the University of Rochester for establishing the Center for Research Computing (CRC). My research simulations heavily depended on parallel computing using Bluehive, a CRC cluster, which accelerated my research progress 50 to 100 times. I cannot imagine how long my Ph.D. research would have lasted without Bluehive.

I would also like to thank all my colleagues at the University of Rochester for their inspiration and collaboration. Among them, I would like to specifically acknowledge: Surjya Ray, Tianqi Wang, Yupeng Jia, Xinye Lou, and Shenqiu Zhang for their friendship and help in my Ph.D. studies; Chaoyi Chen, and Chen-Hsiang Feng for their friendship and help in my job hunting.

Last but not least, I would like to thank my parents and my husband Ashay Narsale, who gave me courage and support during my Ph.D. studies. I would also thank Mrs. Wang Sufen for taking care of me during my thesis preparation.

Abstract

Wireless sensor networks are gaining increasing attention in both industry and academia. One of the main issues that prevents the ubiquitous use of wireless sensor networks is that wireless sensors typically have limited energy supplies. Hence, it is of great importance to save energy and prolong the network lifetime while maintaining the application's Quality of Service (QoS).

For wireless sensors, idle listening consumes power similar to transmitting and receiving data. Therefore, it is beneficial to reduce idle listening by making the sensors sleep whenever possible. My research explores the benefits and limitations of sensor sleeping strategies, conducted at individual layers and at multiple layers of the protocol stack.

I first explore sleeping strategies at different layers. For sleeping at the routing layer, I propose a sleeping scheme for the Directed Diffusion protocol. For sleeping at the MAC layer, I compare the performance of the non-sleeping IEEE 802.11 protocol with S-MAC using different duty cycles. For sleeping at the application layer, I first propose a general 4-layer model to describe the application's QoS requirements, and then I propose a cost function, called Sensor Usage Index (SUI), to determine how often a node should be activated to support the application's QoS. Simulation results for these different sleeping strategies suggest that an adaptive sensor sleeping solution could significantly improve the network lifetime by dynamically controlling the sensor sleeping strategies according to the network conditions and the application requirements.

Such an adaptive sensor sleeping solution needs to quantitatively estimate the performance of different sleeping strategies at every single layer. Therefore, I design a general Markov model to analyze the performance of both synchronized and asynchronous duty-cycled MAC protocols. I also develop a Sleeping Multipath Routing strategy, which utilizes the routing redundancy to trade off between the network lifetime and data reliability.

Finally, I create an adaptive sensor sleeping solution that coordinates between multiple layers based on the Sleeping Multipath Routing protocol and the Markov model for duty-cycled MAC protocols. Simulation results show that the adaptive sensor sleeping solution greatly prolongs the network lifetime under varying application reliability requirements and varying network conditions.

Contents

Foreword **1**

1 Introduction **2**

1.1 Wireless Sensor Networks 2

1.1.1 Applications 3

1.1.2 Challenges 4

1.2 Motivation and Goal 4

1.3 Contributions 6

1.4 Dissertation Organization 8

2 Related Work **9**

2.1 Layered Sleeping Strategies 9

2.1.1 Application Layer Sensor Sleeping 9

2.1.2 Routing Layer Sensor Sleeping 10

2.1.3 MAC Layer Sensor Sleeping 10

2.2 Modeling and Performance Analysis of MAC Protocols 11

2.3 Multipath Routing and Reliability 13

2.4 Modeling Application QoS Requirements 14

2.5 Sensor Selection 16

3 A Better Choice for Sensor Sleeping **18**

3.1 Sleeping at Different Layers Individually and Simultaneously 19

3.1.1 Sleeping at the Routing Layer 20

3.1.2 Sleeping at the MAC Layer 23

3.1.3 Sleeping at Both Routing and MAC Layers 24

3.2 Simulations and Discussions 24
3.2.1 Performance of Single Layer Sleeping 26
3.2.2 Performance Comparisons of Individual Layer Sleeping 27
3.2.3 Performance of Sleeping at Both Layers 32
3.3 Summary 38

4 A General Method to Analyze the Performance of Duty-cycled MAC Protocols 40
4.1 General Markov Models for Queueing Behavior of Duty-cycled Nodes . 42
4.1.1 Duty-cycled Nodes Without Retransmissions 42
4.1.2 Duty-cycled Nodes With Retransmissions 45
4.2 A General Methodology of Protocol-specific Analysis 48
4.2.1 System Model 50
4.2.2 Throughput Analysis 50
4.2.3 Delay Analysis 50
4.2.4 Energy Consumption Analysis 52
4.3 Summary 53

5 Performance Analysis of a Synchronized MAC Protocol - S-MAC 54
5.1 S-MAC Overview 54
5.2 Performance Analysis of S-MAC Without Retransmissions 56
5.2.1 Throughput Analysis 56
5.2.2 Delay Analysis 57
5.2.3 Energy Consumption Analysis 57
5.2.4 Model Validations 62
5.3 Performance Analysis of S-MAC With Retransmissions in Multihop Networks 68
5.3.1 Throughput Analysis 68
5.3.2 Delay Analysis 74
5.3.3 Energy Consumption Analysis 74
5.3.4 Model Validations and Results 74
5.4 S-MAC Optimizations 84
5.5 Discussion 85

5.6 Summary 86

6 Performance Analysis of an Asynchronous MAC Protocol - X-MAC 88

6.1 X-MAC Overview 88

6.2 Performance Analysis of X-MAC 90

6.2.1 Throughput Analysis 90

6.2.2 Delay Analysis 93

6.2.3 Energy Consumption Analysis 93

6.3 Model Validations 95

6.4 X-MAC Optimizations 101

6.5 Summary 101

7 Sleeping Multipath Routing: A Trade-off Between Reliability and Lifetime in Wireless Sensor Networks 103

7.1 Reliability vs. Lifetime 104

7.1.1 Reliability Support Using Multipath Routing and FEC Coding . 104

7.1.2 Sleeping Multipath Routing 105

7.2 Sleeping Multipath Routing: An Example Based on Directed Diffusion 108

7.2.1 Directed Diffusion and Sleeping Directed Diffusion 108

7.2.2 Sleeping Multipath Routing: Disjoint Path Discovery 109

7.2.3 Sleeping Multipath Routing: Multipath Selection 110

7.2.4 Sleeping Multipath Routing: Reliability Guarantee 111

7.3 Simulation Results 111

7.3.1 Varying the Reliability Requirement 112

7.3.2 Varying the Node Density 113

7.4 Summary 113

8 Adaptive Sensor Sleeping 115

8.1 Overview of Sleeping Multipath Directed Diffusion and S-MAC 116

8.1.1 Sleeping Multipath Directed Diffusion 116

8.1.2 S-MAC 118

8.2 Information Sharing Between Layers 119

8.2.1 Information Sharing Between the MAC and Routing Layers . . 119

8.2.2 Information Sharing Between the Routing and Upper Layers . . 120

8.3 Differentiated Packet Delivery 121
8.4 Timer Design 122
8.4.1 Timer Design According to the Duty Cycle of S-MAC 123
8.4.2 Timer Design According to the Markov Model of S-MAC . . . 123
8.5 Adaptive Sensor Sleeping Simulation Results 128
8.5.1 Varying Reliability Requirements 128
8.5.2 Changing Network Conditions 129
8.6 Summary 131

9 A General Sensor Selection Model to Increase Network Lifetime with QoS Support 132
9.1 QoS Description Model and Cost Function 134
9.1.1 4-Layer QoS Description Model 134
9.1.2 Sensor Selection Scheme 140
9.1.3 Sensor Selection Process 144
9.2 Simulation Results 144
9.2.1 Case Study: Warehouse Monitoring 146
9.2.2 General Scenarios 150
9.3 Summary 153

10 Conclusions and Future Research 155
10.1 Summary of Contributions 155
10.2 Future Research Directions 156
10.2.1 Adaptive Sensor Sleeping with Smart Source Node Selection . . 157
10.2.2 Implementation of Adaptive Sensor Sleeping Solution 157
10.2.3 Extending Adaptive Sensor Sleeping Solution to Other Protocols 158

Bibliography 159

List of Tables

4.1 Notations . . . 43

5.1 Fitted Curves . . . 72

7.1 Value Pairs of α and x_α . . . 105

8.1 Estimates of T_1 When $hops = 2$ . . . 126
8.2 Estimates of T_2 When $hops = 2$ . . . 127
8.3 Estimates of T_3 When $hops = 2$ . . . 128

9.1 CC2420 Transceiver Parameters [87] . . . 146
9.2 Sensors in Warehouse Monitoring System [77][78][79] . . . 147

List of Figures

3.1 Time sequences of path establishment in sleeping Directed Diffusion. . 22
3.2 Performance of sleeping Directed Diffusion and SMAC. 26
3.3 Performance comparison under varying node density. 29
3.4 Performance comparison under varying network scale. 30
3.5 Performance comparison under varying number of source nodes. 31
3.6 Performance comparison under varying application data rate. 33
3.7 Performance of multi-layer sleeping with/without cross-layer coordination under varying node density. 35
3.8 Performance of multi-layer sleeping with/without cross-layer coordination under varying network scale. 36
3.9 Performance of multi-layer sleeping with/without cross-layer coordination under varying number of sources. 37
3.10 Performance of multi-layer sleeping with/without cross-layer coordination under vary data rate. 38

4.1 Markov model for queueing behavior of duty-cycled nodes without retransmissions. 44
4.2 Markov model for queueing behavior of duty-cycled nodes with retransmissions. 46

5.1 Sleep-awake cycle of S-MAC. 55
5.2 Determining (p, π_0) using $\pi_0 = f(p)$ and $p = g(\pi_0)$ for S-MAC. 57
5.3 S-MAC performance with different numbers of nodes in the network. . 63
5.4 S-MAC performance with different data arrival rates. 64
5.5 S-MAC performance with different contention window sizes. 65
5.6 Examples of model validations. 76

5.7 S-MAC performance with different node density in multihop networks. 79
5.8 S-MAC performance with different data arrival rates in multihop networks. 81
5.9 S-MAC performance with different contention window sizes in multihop networks. 83
5.10 S-MAC performance optimization under different contention window sizes and duty cycles. 85

6.1 X-MAC timeline. 90
6.2 X-MAC performance with different cycle lengths. 97
6.3 X-MAC performance with different numbers of nodes in the network. . 98
6.4 X-MAC performance with different data arrival rates. 99
6.5 X-MAC performance optimization under different cycle lengths. 102

7.1 Revisit of Sleeping Directed Diffusion [26]. 109
7.2 Lifetime of Sleeping Multipath Directed Diffusion (SMDD) and no sleeping Multipath Directed Diffusion (MDD) under varying reliability requirements. 113
7.3 Lifetime of Sleeping Multipath Directed Diffusion (SMDD) under varying node density. 114

8.1 Revisit of Sleeping Multipath Directed Diffusion. 116
8.2 S-MAC queue vs. differentiated packet delivery. 122
8.3 Lifetime comparison of adaptive sensor sleeping and non-adaptive sensor sleeping when the reliability requirement changes. 130
8.4 Lifetime comparison of adaptive sensor sleeping and non-adaptive sensor sleeping when the number of nodes in the network changes. 131

9.1 4-layer QoS description model. 134
9.2 QoS description of the warehouse monitoring application. 136
9.3 Sensor set selection. 143
9.4 Sensor selection process at the base station. 145
9.5 Network lifetime comparisons for warehouse monitoring. 147
9.6 Use of sensors in Min P selection. 148

9.7 Use of sensors in $1/E$ selection. 149
9.8 Use of sensors in SUI selection. 149
9.9 Network lifetime comparisons in homogeneous networks. 152
9.10 Network lifetime comparisons in heterogeneous networks. 153

Foreword

I conducted all of the research work from Chapter 3 to Chapter 9. Each of these chapters has been published as described below. For all of these publications, I am the primary author, and I collaborated with Dr. Wendi Heinzelman.

The work described in Chapter 3 has been published in the proceedings of the European Conference on Wireless Sensor Networks (EWSN), 2009.

The work described in Chapter 4 has been accepted for publication in IEEE Transactions on Mobile Computing, 2011.

The work described in Chapter 5 has been published in the proceedings of the International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2009.

The work described in Chapter 6 has been published in the proceedings of the Global Communication Conference (GlobeCom), 2010.

The work described in Chapter 7 has been accepted for publication in the proceeding of Global Communication Conference (GlobeCom), 2011.

The work described in Chapter 8 is currently under preparation for submission.

The work described in Chapter 9 has been published in the proceedings of the ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), 2008.

Chapter 1

Introduction

Wireless sensor networks have been available for decades. However, until recently, wireless sensor networks were not ready for practical use due to their high cost. With the continued development of semiconductor technology, techniques to reduce energy dissipation, and System-on-a-Chip (SoC) integration technology in the past 10-20 years, wireless sensor networks have now become one of the first real-world examples of pervasive computing [1]. This dissertation addresses one of the main challenges of the ubiquitous use of wireless sensor networks, limited energy, by optimizing the sleeping strategies of networked wireless sensors.

1.1 Wireless Sensor Networks

A wireless sensor network is a network that consists of spatially distributed autonomous sensors, working cooperatively to monitor environmental conditions. Typically, a wireless sensor is equipped with one or more sensors, a radio transceiver to support wireless communication, a small micro-controller to process data and for protocol signaling, and an energy source, usually a battery, to provide power to the sensors, the transceiver, and the micro-controller. Wireless sensors can form a network with various topologies, such as ad-hoc networks, clustered networks, or centralized networks.

Wireless sensor networks are gaining increasing attention for practical use ranging from military surveillance to civil monitoring due to their low cost, ease of deployment and good coverage of the monitored area. In academia, wireless sensor networks also attract much research interest since emerging applications push wireless sensor networks to be ever more durable, dynamic and reliable.

1.1.1 Applications

Wireless sensor networks are being used in various areas, ranging from military and industry to medicine and health care. Typical applications of wireless sensor networks include:

(1) Military surveillance. Wireless sensor networks can help achieve effective battlefield situational awareness. For example, one idea is to use an aircraft to scatter a large number of wireless sensor nodes in a vast battlefield area. The wireless sensors create a network, which collects battlefield information and relays it to the control center.

(2) Industrial monitoring. There are many applications that require the monitoring of goods. For example, the food industry uses wireless sensor networks to prevent repeats of the contamination of the food supply chain [2]. As another example, logistical applications benefit from wireless sensor networks to simplify supply chain management and reduce cost [3].

(3) Habitat monitoring. Sensor networks have found varied applications in the area of habitat monitoring. In 2004, Princeton University launched a project called ZebraNet [4]. The project deployed wireless sensor nodes on wild zebras in Kenya to monitor the animal migrations, inter-species interactions, and nocturnal behavior. A bird observation project on Great Duck island also used wireless sensor networks to measure the humidity, pressure, temperature, etc., in the birds' burrows in order to understand the birds' behavior [5].

(4) Precision agriculture. Wireless sensor networks can be used to monitor the micro-climate such as temperature, moisture, PH of water-nutrient mixture, and so on, in a crop field to enable more efficient and precise field maintenance [6].

(5) Emergency rescue. Wireless sensor networks are also used in emergency search and rescue in cases such as fire [7], avalanche [8], and earthquake [9] to help locate the survivors.

(6) Health monitoring. Body sensor networks have been used in mobile health monitoring in Europe and Australia [10]. A patients' biosignals are measured by means

of body worn sensors that communicate wirelessly with a handheld device. Alarms and biosignals can be transmitted over wireless communication links to a remote location, and a health professional can view the biosignal data via a web application and take appropriate actions.

1.1.2 Challenges

There are many issues preventing the ubiquitous use of wireless sensor networks due to the characteristics of the sensor nodes or the wireless networks.

First of all, wireless sensors have limited energy supply, as they are usually battery-powered. However, it is oftentimes difficult or impractical to change the battery as the sensors may be deployed in large scale [11] or in an unsafe environment [5]. Therefore, it is important to save energy at each sensor while supporting the application Quality of Service (QoS) requirements. This challenge motivates energy-efficient protocol design and research in energy-efficient hardware.

Moreover, wireless sensors usually have limited storage and computation capability. Therefore, sensors may not be able to do sophisticated data processing and hence light-weight protocols are required.

Furthermore, wireless sensor networks have to be fault-tolerant, as they may experience challenges such as harsh environmental conditions, node failure, dynamic topology, etc. To meet the application requirements, protocols designed for wireless sensor networks are required to achieve a certain level of robustness. For example, using more than one routing path could avoid communication failure resulting from a single node failure on a routing path or poor communication quality on a specific link.

Wireless sensor networks also face challenges in security and privacy, quality of service, scalability, and real-time response, among others.

1.2 Motivation and Goal

Wireless sensors have limited energy supply since they are usually battery-powered, and oftentimes it is impractical or inconvenient to recharge the sensors manually. Hence, it is critical to employ energy-saving techniques so as to support an application for extended periods of time. Among the existing energy-saving strategies, e.g., flooding avoidance [12][13], traffic balancing using cost functions [14][15], and data fusion [16], putting sensors into the sleep state is the most widely-used and cost-effective way to prolong the application lifetime. By turning off the radio of a sensor at appropriate times, various sleeping schemes [17][18][19][20][21][22][23][24] aim to cut down on "idle listening," which in wireless sensor networks provides little benefit yet consumes almost as much power as transmitting or receiving data [25].

To prolong the network lifetime, different layers in the protocol stack can put redundant sensors to sleep according to their own benefit. For example, the application layer can select and activate a subset of the source nodes to support the application QoS and put the rest of the source nodes to sleep, saving their energy for later use. The routing layer, on the other hand, can either use topology control protocols to limit the number of activated routers in the network, or it can put sensors to sleep when the routing protocol determines that they are not involved in the data delivery. The MAC layer, at the same time, can utilize sleep-awake cycles to reduce idle listening.

As sleeping can be implemented at the application layer, at the network layer, and at the MAC layer individually, questions arise as to which layer provides the most benefit for determining when a sensor should sleep, under what conditions should each sleeping scheme be chosen, and whether cross-layer coordination is needed to obtain further improvement when sleeping schemes are employed at different layers simultaneously. The answers to these questions not only guide us in the implementation of practical sensor networks, but they also motivate us to propose an adaptive sensor sleeping solution that can dynamically control the sensor sleeping strategy according to the current network conditions and application requirements, with cross-layer sleeping management. Using the adaptive sensor sleeping solution, we expect that the sensor network will have a longer network lifetime with QoS support.

To determine the best approach to adapt the sensor sleeping strategy, the following knowledge must be obtained. First, what are the benefits and limitations of different sleeping strategies at different layers? Second, how can we estimate the performance of sleeping strategies at each layer quantitatively? Third, what coordination is needed to obtain a longer lifetime when sensors can sleep at multiple layers? Finally, how can we make the dynamic decision on how to utilize sensor sleeping? The goal of our adaptive sensor sleeping solution is to optimize the network lifetime and provide flexibility to trade off network lifetime with data reliability.

1.3 Contributions

This dissertation aims to understand the benefits and limitations of various sleeping strategies in wireless sensor networks, and to provide an adaptive sensor sleeping solution to dynamically decide the appropriate sleeping strategy according to the network conditions and the application requirements. The contributions of the research include:

(1) Studying the performance of various sensor sleeping strategies at different single layers and at multiple layers, and understanding their benefits and limitations under various network conditions and application requirements [26]. Using simulations we show that dynamically choosing an appropriate sensor sleeping strategy is promising in further prolonging the network lifetime. Also, proper coordination between different layers is found to be necessary in multi-layer sleeping strategies.

(2) Proposing a sleeping scheme for Directed Diffusion, a popular routing protocol for wireless sensor networks [26]. Our proposed sleeping scheme significantly improves the application lifetime by allowing the nodes that are not involved in the data transmission to sleep periodically.

(3) Generalizing an approach to evaluate the performance of any duty-cycled MAC protocol by proposing (a) Markov models to describe the general queuing behavior of duty-cycled MAC protocols, and (b) a general methodology to handle the protocol-specific performance analysis based on the proposed Markov models [27]. Our approach works for both synchronized and asynchronous duty-cycled MAC protocols, and it provides comprehensive performance metrics, including throughput, delay, and energy consumption. Moreover, our proposed Markov models as well as the performance analysis method can be used to optimize the protocol parameters in order to achieve more complex performance goals.

(4) Analyzing and optimizing the performance of S-MAC, a synchronized duty-cycled MAC protocol for wireless sensor networks, using the proposed Markov models and the performance analysis method [27][28]. Since S-MAC is the first and the most basic duty-cycled MAC protocol for wireless sensor networks, it is oftentimes used as

a baseline in performance comparisons with newly proposed duty-cycled MAC protocols. Hence, future research on duty-cycled MAC protocols can benefit from the convenience and flexibility provided by our Markov models for S-MAC to avoid massive simulations.

(5) Analyzing and optimizing the performance of X-MAC, an asynchronous duty-cycled MAC protocol for wireless sensor networks, using the proposed Markov model and the performance analysis method [27][29]. This is the first theoretical performance analysis for X-MAC.

(6) Proposing Sleeping Multipath Routing to significantly prolong the network lifetime while maintaining a certain reliability performance by putting sensors to sleep at the routing layer [30]. Sleeping Multipath Routing can also be used to trade off reliability for lifetime.

(7) Proposing an adaptive sensor sleeping solution for Sleeping Multipath Directed Diffusion and S-MAC. The solution is based on the design of Sleeping Multipath Routing and the Markov model for S-MAC performance analysis. The adaptive sensor sleeping solution prolongs the network lifetime with reliability guarantee under varying reliability requirements and network conditions.

(8) Proposing a general model to describe an application's QoS requirements. The model provides (a) a tool for determining all possible sensor sets that can support required QoS and help the sensor selection scheme to prolong the network lifetime with QoS support, and (b) flexibility to support intelligent applications, which may have dynamic QoS requirements. Our model may greatly reduce the computational complexity compared to an exhaustive search.

(9) Proposing a cost function called SUI (Sensor Usage Index) and a set of sensor selection criteria according to SUI to put redundant source nodes to sleep [31]. SUI is the first cost function that considers a sensor's diversity in application contribution, remaining energy, and power, simultaneously. It greatly prolongs the network lifetime, especially in heterogeneous networks.

1.4 Dissertation Organization

This dissertation is organized as follows. Chapter 2 elaborates on the related work in the area of layered sleeping strategies, MAC protocol modeling and analysis, multipath routing, QoS description models, and cost functions for sensor selection. Chapter 3 proposes a sleeping scheme for Directed Diffusion, and studies the benefits and limitations of various sleeping strategies at a single layer and at multiple layers. Chapter 4 proposes a general method to analyze the performance of duty-cycled MAC protocols. Based on this general method, Chapter 5 analyzes the performance of S-MAC, a synchronized duty-cycled MAC protocol, without retransmissions in fully-connected networks and with retransmission in multihop networks. Similarly, Chapter 6 analyzes and optimizes the performance of X-MAC, an asynchronous duty-cycled MAC protocol. Chapter 7 proposes Sleeping Multipath Routing to trade off network lifetime with data reliability. Chapter 8 provides an adaptive sensor sleeping solution for Sleeping Multipath Directed Diffusion and S-MAC under varying reliability requirements and network conditions. Chapter 9 proposes a general QoS description model and a new cost function SUI for sensor selection to prolong the network lifetime. Chapter 10 finally concludes this dissertation.

Chapter 2

Related Work

2.1 Layered Sleeping Strategies

Since wireless sensor networks are energy constrained, and idle listening is energy intensive, to save the most energy, it is intuitive to make sensors sleep whenever possible. This implies that (1) redundant source nodes may sleep when their sensed data is not required by the sink, (2) redundant routing nodes may sleep when they have no data to relay, and (3) source nodes and routing nodes that are involved in the data delivery may also sleep when it is not their turn to transmit or receive data. Correspondingly, existing sleeping schemes are implemented in such a way that different layers perform these different tasks mentioned above.

2.1.1 Application Layer Sensor Sleeping

In application layer sensor sleeping, the application layer selects and activates only some of the source nodes to reduce energy drain while maintaining desired QoS [17][32]. For example, in the application of target tracking [17], all sensors within a certain distance of the target could be source nodes, but only a few are finally activated by the application layer to achieve low distortion and reduce the energy consumption. As another example, for a coverage application [32], the application layer may use various cost functions to describe each sensor's monitoring redundancy, and then the application can choose a subset of all the sensors, based on each sensor's cost and coverage, to support the required coverage while minimizing the cost. Those sensors that could provide data to the application but are not selected by the application layer are put to sleep to save energy for later use.

2.1.2 Routing Layer Sensor Sleeping

Routing nodes, on the other hand, are usually selected and activated by the network layer in two ways. The first approach is called topology control [18][19], which aims to establish a routing backbone to guarantee the network connectivity, and consequently non-backbone nodes can sleep. Topology control also updates the backbone periodically according to the remaining energy of each sensor to balance the energy consumption. For instance, GAF [18] is a topology control protocol that divides the network into virtual grids with only one sensor in each grid activated at any time, constituting a backbone network to route all the data. Topology control is often used in large scale, dense networks where many nodes provide redundant routes. Hence, making all of them except the activated sensors sleep can significantly extend the application lifetime.

The second approach for making sensors sleep at the network layer is through routing protocols. Routing protocols [20][21][24] update routes periodically, and save energy by turning off the routing nodes that are not involved in the data delivery. For example, the technique described in [20] puts sensors to sleep by monitoring routing control messages and data transmissions so that only useful routers are kept active. The technique described in [21] also utilizes the routing decisions so that sensor nodes do not wake up when they are not part of a routing path. Compared to topology control, those routing protocols with sleeping do not waste the energy of non-used routing nodes, and they are also suitable for sparse or not large scale networks where topology control performs poorly.

2.1.3 MAC Layer Sensor Sleeping

Once source nodes and routing paths are determined, the MAC layer can put sensors into sleep-awake cycles [22][23] so that idle-listening can be further reduced due to the fact that the traffic load in wireless sensor networks is usually not very high [22]. For example, S-MAC [22] makes each sensor broadcast its sleep-awake schedule once in a while, so that its neighboring sensors can hear it and synchronize their schedules with it. Once sensors are synchronized, they use RTS/CTS/DATA/ACK sequences to communicate during the awake period of each cycle. B-MAC [23], using periodic low power listening, forces a sensor to sleep if nothing is detected on the media in the sensor's awake duration. However, the benefit of longer application lifetime provided by MAC layer sleeping accompanies lower throughput and higher latency.

Among duty-cycled MAC protocols, some of them synchronize all the nodes in the network, so that the nodes sleep and wake up at the same time. Synchronization guarantees that the receiver is ready to receive when a sender wakes up and has a packet to send, and hence it improves the communication efficiency. However, synchronization requires overhead at each node to exchange sleep-wake-up schedules with the nodes' neighbors. This overhead can be significant when the data traffic is light in the network. Example synchronized duty-cycled MAC protocols include S-MAC [22] and T-MAC [33].

On the other hand, some of the duty-cycled MAC protocols are asynchronous. Asynchronous MAC protocols also put sensors to sleep periodically. However, every node has an arbitrary offset to start its sleep-wake-up cycles. In this case, the synchronization overhead is removed, but a sender with a packet to send may have to delay the transmission until the receiver wakes up. Example asynchronous duty-cycled MAC protocols include X-MAC [34] and Spec-MAC [35].

2.2 Modeling and Performance Analysis of MAC Protocols

Much work has been done to evaluate the performance of various MAC protocols for wireless sensor networks. Most of the performance evaluations [36][37][38] are obtained from simulations. However, simulations are usually time consuming and require a large number of runs to obtain statistically significant results. Some other work implemented MAC protocols on motes and obtained their performance using field measurements [39][40]. However, constrained by time, space and available resources, field measurements are oftentimes a case study, from which it is difficult to draw general or quantitative conclusions on the performance of a protocol. Therefore, analytical models are needed to provide insight into the performance of MAC protocols.

Analytical models have been proposed to evaluate the performance of a specific MAC protocol. For example, Bianchi proposed a Markov model to analyze the saturation throughput of IEEE 802.11 with unlimited retransmissions [41]. Pollin et al. proposed a Markov model to analyze the performance of slotted IEEE 802.15.4 [42]. Buratii et al. analyzed the performance of non beacon-enabled IEEE 802.15.4. Zhang et al. created a Markov model to analyze the network throughput, power consumption and packet service delay of S-MAC in a single hop network [43]. However, their model cannot derive the packet queuing delay, which has a significant impact on the packet latency. Zhang et al. proposed an analytical model to evaluate the performance of O-MAC [37]. Although these models can estimate the performance of a specific protocol, they are fundamentally different and cannot be generalized as different protocols have different media access rules. Hence, the application of these models is limited.

Some other work focused on analyzing a specific performance metric, for example, delay, or energy consumption, for a specific MAC protocol or a series of MAC protocols that have the same characteristics. For example, Wang et al. analyzed the distribution of the end-to-end delay for CSMA/CA based MAC protocols in wireless sensor networks [44]. Fischione et al. modeled the packet delay in un-slotted IEEE 802.15.4 networks [42]. Wang et al. analyzed the data delivery delay in acoustic sensor networks using queuing theory [45]. Rousselot et al. calculated the lower bound of power consumption for a set of scheduled access and random access MAC protocols for wireless sensor networks [46]. Although some of these works made a general conclusion among certain MAC protocols, their performance evaluations are constrained for only one metric of a protocol's performance. Hence, their models cannot be used to determine the trade-offs in different performance metrics.

Luo and et al. created continuous time Markov models and queuing models to analyze the packet loss, delay and power consumption of contention-based MAC protocols with synchronized and asynchronous wake-up patterns [47][48]. However, their Markov models for synchronized wake-up patterns and asynchronous wake-up patterns were different, and hence cannot be generalized. Moreover, their models obtained the stationary probability of the empty queue state assuming the number of contending neighbors of each node was known. In fact, the stationary probability of the empty queue state in return determines the number of contending neighbors of each node in the network. Additionally, their models assume the knowledge of packet transmission rate at each node in a cycle. However, the packet transmission rate at each node is also related to the contention in the network. Hence, the assumption of knowing the number of contending neighbors of each node and the assumption of knowing the packet transmission rate at each node in a cycle are impractical.

In terms of supporting retransmissions, some work has been done to analyze the performance of various MAC protocols using Markov models with retransmissions. For example, Bianchi proposed a Markov model to analyze the saturation throughput of IEEE 802.11 with unlimited retransmissions [41]. Wu et al. modified Bianchi's model to support finite retransmissions [49]. Later on, similar Markov models were proposed to analyze IEEE 802.16 and IEEE 802.15.4 networks with retransmissions [50][51]. However, most of the previous work analyzed the performance of a MAC protocol in fully-connected networks [52][53], since hidden terminals in multihop networks are hard to formalize.

In [54], the authors formalized the throughput of IEEE 802.11 networks with hidden terminals. However, the calculation of the throughput is topology-specific, and hence it is hard to obtain the general performance of a protocol under an arbitrary topology. The authors in [55] also proposed a method to estimate the number of nodes in the hidden area of a node, and then use this value to determine the throughput of IEEE 802.11. However, the number of hidden nodes we obtained using the calculation proposed in [55] is different from the results obtained in [55], and this estimate does not provide accurate results in our Markov model.

2.3 Multipath Routing and Reliability

Reliability is an issue in providing desirable quality of service (QoS) for applications that are using wireless sensor networks. For example, military surveillance applications may require that the data delivery ratio be above a certain threshold, while applications that monitor a structure's vibration may need vibration measurements no less than every 5 minutes. Since wireless channels are error-prone, and multihop communications are oftentimes necessary in various applications, data reliability over multihop transmissions may be significantly degraded. To solve this problem, Forward Error Correction (FEC) codes and retransmission techniques are used by the MAC protocols to make each hop more reliable. Multipath routing [56][57][58] is also widely used to increase the likelihood of reliable data delivery by sending multiple copies of data along different paths [56]. Moreover, Forward Error Correction codes can be combined with multipath routing protocols to enhance the reliability [56]. Specifically, suppose there are N disjoint paths from the source node to the sink node. A packet can be coded into k subpackets with $N - k$ redundant subpackets, with each subpacket delivered via a different path. The sink node can reconstruct the packet as long as at least K out of N paths deliver subpackets successfully.

Multipath routing can greatly improve the reliability in wireless sensor networks. However, multipath routing also requires more nodes to be involved in the data delivery, implying more energy consumption and thus a shorter network lifetime. To the best of our knowledge, sensor sleeping has not been introduced into multipath routing protocols to save energy and prolong the network lifetime. We therefore propose Sleeping Multipath Routing in Chapter 7, which discovers disjoint multiple paths in a network, selects the minimum number of disjoint paths to meet the reliability requirement, and puts the rest of the nodes in the network to sleep. When the current disjoint paths deplete their energy, Sleeping Multipath Routing discovers new disjoint paths and selects some of them to support the reliability, until no set of paths can be found to achieve the reliability requirement.

2.4 Modeling Application QoS Requirements

Wireless sensor networks have been proposed for use in various fields, including military, industry, agriculture, and medical monitoring, due to their beneficial characteristics of low power, low cost, ease of deployment and ability to provide continuous monitoring. For small range applications, like indoor security, clinical monitoring, and aircraft recording, single-hop centralized networks work efficiently with small overhead and no collisions (assuming data scheduling is used at the MAC layer). However, challenges arise in these networks as such monitoring applications tend to have sophisticated QoS requirements, so that discovering all possible sensor sets that support QoS is not an easy task.

Extensive research [15][59][60][61][62][63][64][65][66][67][68][69] has been done on routing and sensor selection techniques to support a particular application, such as coverage [59][62][69], gas detection [64], and vehicle guidance [68]. However, different applications have different measures of QoS; hence much past work [59][62][64][65][68][69] has looked at application-specific strategies to schedule the sensors. However, these schemes do not easily generalize, making it difficult to implement new applications for wireless sensor networks. Therefore, a general model to describe an application's QoS requirements is desirable, as this will provide a tool for determining all possible sensor sets that can support the required QoS and help the sensor selection scheme to prolong the network lifetime with QoS support.

Moreover, applications for wireless sensor networks are evolving from supporting simple monitoring tasks to supporting sophisticated combinations of monitoring and decision-making [64]. For example, a coverage application may no longer be interested in prolonging the lifetime of 100% coverage only, but it may also try to support different QoS requirements on percentage of coverage at different points in time, in different locations in the area being monitored (e.g., room), or using different sensing devices. By either analyzing the sensed data, or by receiving instructions from users, the base station can inform the sensors of any transition of the application's QoS requirements, and the base station can decide how to activate and schedule the sensors to satisfy the current QoS requirements. Hence, a general model for quantifying QoS should also provide the flexibility to support intelligent applications, which may have dynamic QoS requirements.

Furthermore, computational complexity is also an important consideration in obtaining all possible sensor sets under different QoS requirements. It is not desirable, or even practical, to exhaustively search every possible combination of sensors to determine the sensor sets that meet the application QoS requirements. Consequently, it is desirable to develop a general procedure that could be used in diverse applications, to reduce the searching range for finding sensor sets that support QoS.

2.5 Sensor Selection

Wireless sensor networks oftentimes have redundancies, as multiple sensors can have overlapping contributions to the application's functionality. Hence, in a particular scenario, more than one sensor set may provide QoS support to an application. A sensor selection scheme is responsible for choosing one of these valid sets as the final activated sensor set, so as to prolong the network lifetime with QoS support.

The simplest sensor selection scheme is to choose the final sensor set randomly from among the sets that meet application QoS requirements. However, random selection cannot optimize the network lifetime by selectively using the sensors. Another scheme [61] [66] is to select the sensor set that consumes minimum power in each round of transmission. When more than one sensor set have the same total power consumption, the final sensor set is randomly selected from among those with minimum power. This scheme conserves energy for the entire network, but it might not optimize energy usage for individual sensors. Some important sensors used to support the application's functionality might die early if they happen to have small power consumption and low remaining energy. The third sensor selection scheme [15][60][67] takes into account the idea that we should avoid selecting sensors with low remaining energy, if possible. Hence $1/E$, where E is a sensor's remaining energy, is proposed as the sensor's cost function. In this selection scheme, a smaller cost means that it is more preferable that the sensor be used. Hence the sensor set that has the minimum total cost among all possible sensor sets will be selected. However, the basic assumption behind the $1/E$ scheme is that all the sensors in the network are homogeneous in power consumption. When this is not the case, a sensor's remaining lifetime depends on both its remaining energy as well as its power consumption. A sensor with large remaining energy might die soon if it has very large power consumption, and thus this sensor needs to be used conservatively.

In wireless sensor networks, sensors consume energy both in sensing data and in transmitting the sensed data to a base station. The power consumption for transmitting data is an exponential function of the distance from the sensor to the base station (assuming transmission power control is used), while the power consumption for sensing data is determined by the type of sensor (e.g., thermometer, pressure sensor, microphone, camera, etc.) as well as the sensing technology. Hence, different sensors may have very different power consumptions. As a result, for many applications that involve heterogeneous sensors (e.g., an avalanche rescue project [8] uses oximeters, oxygen sensors and accelerometers; a body sensor network [63] uses ECG sensors, SpO2 sensors, accelerometers, temperature and humidity sensors), the existing sensor selection schemes may not achieve desirable network lifetime, since neither the sensor's power consumption nor the sensor's remaining energy can be used individually to optimally prolong the network lifetime with QoS support.

Chapter 3

A Better Choice for Sensor Sleeping

As protocols at different stack layers can make sensors sleep, this chapter tries to answer the following questions. (1) At which layer should sleeping be used in order to extend application lifetime for a specific network and application scenario? (2) Is some sort of coordination needed so that sleeping at multiple layers gains more than simply sleeping at a single layer? In this chapter, we first propose a sleeping scheme for Directed Diffusion [13], which significantly improves the application lifetime by allowing nodes not involved in the transmission of data to sleep periodically. Then we compare sleeping techniques employed by routing protocols and MAC protocols under various node densities, different network scales, diverse numbers of source nodes and varying application data rates. We also examine the performance of making sensors sleep at the routing and MAC layers simultaneously, with and without cross-layer coordination. Application layer sleeping schemes are currently not considered to avoid any application-specific conclusions.

The rest of the chapter is organized as follows. Section 3.1 proposes our approach, including a new Directed Diffusion protocol with sleeping, to compare the performance of sleeping at the routing and MAC layers, individually and simultaneously. Section 3.2 analyzes the simulation results obtained from various network scenarios under different application requirements. Section 3.3 summarizes the chapter.

3.1 Sleeping at Different Layers Individually and Simultaneously

As wireless sensing applications operate in diverse networking environments and have a range of QoS requirements, source node selection at the application layer is always application-specific, and hence it is hard to generalize its performance. Therefore, to avoid any application-specific conclusions, in this chapter we choose a general query-based application, and we focus on more general sleeping strategies at the network and MAC layers. Specifically, we look into sleeping schemes implemented in the routing and MAC protocols, defined as “routing layer sleeping” and “MAC layer sleeping”, respectively, throughout this chapter. Topology control (done by the network layer) is currently not investigated. We make conclusions by comparing the performance of (1) a non-sleeping routing protocol with a non-sleeping MAC protocol, (2) a non-sleeping routing protocol with a sleeping MAC protocol, (3) a sleeping routing protocol with a non-sleeping MAC protocol, and (4) a sleeping routing protocol with a sleeping MAC protocol.

For the non-sleeping routing protocol, we choose Directed Diffusion [13], as it is specially designed for data-centric sensor networks and widely accepted as a typical routing protocol for wireless sensor networks. However, Directed Diffusion does not include any sleeping strategy. We, therefore, propose an improved Directed Diffusion with sleeping as our sleeping routing protocol (see details in section 3.1.1). The sleeping Directed Diffusion performs exactly the same as the original Directed Diffusion in terms of source node discovery and routing path establishment and maintenance, but sleeping Directed Diffusion allows routing nodes to sleep during the interval between two successive routing updates, if they are not reinforced by the sink to relay data. Therefore, the application lifetime may be prolonged by getting data from another routing path once the previous path dies due to energy depletion.

For the MAC layer, we use IEEE 802.11 and SMAC as the non-sleeping and sleeping protocols, respectively. SMAC is a typical MAC protocol with duty cycled sleeping for wireless sensor networks. Except for the duty cycle, SMAC is similar to IEEE 802.11. In other words, SMAC with 100% duty cycle has the same performance as IEEE 802.11 under light traffic load. Hence, comparing the performance of the network using IEEE 802.11 and SMAC provides clear insight into the benefits and drawbacks of duty cycling the sensors at the MAC layer.

3.1.1 Sleeping at the Routing Layer

To understand our proposed sleeping Directed Diffusion, we first review the mechanisms of Directed Diffusion, and then we explain our implementation of sleep-awake cycles used with Directed Diffusion.

3.1.1.1 Directed Diffusion

Directed Diffusion [13] includes two phases, an exploratory phase and a reinforcement phase, which together allow a sink node to obtain desirable data from source nodes. The exploratory phase is used to discover data sources at a low data rate, while the reinforcement phase is used to pull down the desirable data at a high data rate.

In the exploratory phase, a sink starts broadcasting INTEREST packets periodically in the network. An INTEREST packet includes a description of desired data attributes and indicates the exploratory rate of this specific data (usually as low as one packet per hundreds of seconds). A node that receives an INTEREST packet floods it to all of its neighbors. Meanwhile, every node maintains a gradient table, caching each distinguishable INTEREST packet in terms of where it came from (previous hop) and what the desired data rate is as the local routing information. Each caching entry is called a gradient, which expires and hence will be removed from the gradient table after a certain time, so as to take care of topology changes or node failures. New gradients will be established by the periodic INTEREST packets flooded in the network. As a result, after some time, all the nodes, including the data sources in the network know the INTEREST, and all possible routing paths are established in a distributed manner by checking gradient information at each node. When a source node receives an INTEREST packet, it broadcasts DATA packets to all its neighbors at the exploratory rate. All the intermediate nodes cache and flood all distinguishable DATA packets and discard duplicate ones (e.g., if the time stamp is very close - less than the interval of two successive high rate DATA packets - to any received DATA packet). Finally, the sink node receives the DATA packets that it is interested in from multiple paths. Then the sink starts the reinforcement phase.

In the reinforcement phase, the sink node has a specific policy to reinforce some of the routing paths to pull down the data from the source nodes at high data rates. The policy in our experiments is to reinforce the neighbor that delivered the exploratory data first. The sink node unicasts a positive reinforcement packet, which is actually an INTEREST packet with high desired data rate (usually tens of times higher than the exploratory rate), to the selected neighbor, and then the selected neighbor forwards this positive reinforcement packet to the next hop, which is chosen by the same policy. When a source node receives a positive reinforcement packet, it starts sending back DATA packets at the requested high data rate, along the path where the positive reinforcement packet came from. Therefore, the sink node finally obtains data at the desired high rate from the reinforced path. In the case of topology changes, node failures or link quality changes, a new path may be positively reinforced. Hence, every sensor periodically checks if a negative reinforcement is needed to slow down the data delivery on the previously reinforced paths.

Directed Diffusion does not allow sensors to sleep. However, not every routing node is involved in the data delivery all the time, as only the least delayed routing path is positively reinforced. Hence, allowing redundant routing nodes to sleep may save other routing paths and contribute to a longer application lifetime when the previous ones run out of energy.

3.1.1.2 Sleeping Directed Diffusion

To make routing nodes sleep when they are not involved in the data delivery, it is necessary to note the importance of INTEREST flooding and exploratory DATA flooding, and figure out the time sequences of path establishment and maintenance in Directed Diffusion.

As mentioned above, in Directed Diffusion there are two critical floodings throughout the network. One is the periodic flooding of an INTEREST packet, initiated by the sink node. All routing nodes need to be awake to forward this packet so that source nodes can finally receive it, and a gradient table can be established to route the DATA packets that are going to follow. The other flooding is exploratory DATA flooding, periodically initiated by a source node. All routing nodes need to be awake to forward those packets so that the sink node can finally receive them and start positive reinforcement

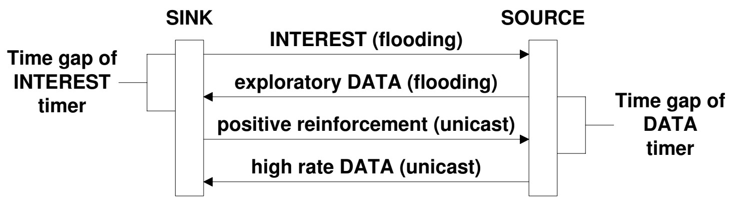

SINK
SOURCE
INTEREST (flooding)
exploratory DATA (flooding)
positive reinforcement (unicast)
high rate DATA (unicast)
Time gap of INTEREST timer
Time gap of DATA timer

Figure 3.1: Time sequences of path establishment in sleeping Directed Diffusion.

based on their arrival times. As a result, for each distinguishable INTEREST (query for different data attributes), we define two timers, INTEREST timer and DATA timer, at each node, for INTEREST flooding and exploratory DATA flooding, respectively. The two timers are scheduled and fire periodically. Specifically, INTEREST timer is scheduled after an INTEREST flooding ends, and fires before the next INTEREST flooding starts, while DATA timer is scheduled after an exploratory DATA flooding ends, and fires before the next exploratory DATA flooding starts. A sensor may sleep when the two timers are pending, but it MUST wake up once either timer fires, and remain awake until the timer is rescheduled.

There is a time gap between when a timer fires and when it is rescheduled. The time gap is used to wait for the response of the corresponding flooding so as to establish a reinforced path from the sink node to the source node. As shown in Fig. 3.1, path establishment starts with INTEREST flooding, followed by the exploratory DATA flooding, which is then followed by positive reinforcement unicasting, and consequently followed by high rate DATA unicasting. Hence, the time gap of the INTEREST timer is used to wait for the exploratory DATA flooding if new source nodes are discovered, while the time gap of the DATA timer is used to wait for the positive reinforcement packets so that one of the source nodes can be reinforced and start delivering high rate data.

If a node does not receive any exploratory DATA packet during the time gap of its INTEREST timer as a result of no source node available or packet loss, the exploratory DATA timer is not initiated. Hence, the node follows the INTEREST timer to sleep (when it is pending) and wake up (when it expires). Otherwise, the exploratory DATA timer is initiated, and it is scheduled periodically according to the exploratory rate. If the node receives a positive reinforcement packet during the time gap of its DATA timer, the node is located on the reinforced path (involved in the data delivery), and thus cannot sleep until it is negatively reinforced or it runs out of energy. If the node does not receive any positive reinforcement packet during the time gap of its DATA timer, the node goes to sleep when both of the timers are pending (no flooding is going on), and wakes up whenever at least one of the timers expires (at least one flooding is going on).

3.1.2 Sleeping at the MAC Layer

SMAC is a MAC protocol explicitly proposed for wireless sensor networks, with reducing energy consumption as its primary goal [22]. It introduces periodic sleep-awake cycles for each node to eliminate idle-listening. Used for transmitting and receiving data, the awake period of a cycle has a fixed length, which is determined by the physical layer and MAC layer parameters. The sleeping period of a cycle, instead, can be set longer or shorter, which influences the power consumption of SMAC, as well as the latency incurred by sleeping. Hence, variations in duty cycle, which is defined as the ratio of the awake period to a complete sleep and awake cycle, leads to corresponding variations in the performance of SMAC.

For the convenience of communication, and to reduce the control overhead, SMAC tries to synchronize every node, so that nodes sleep and wake up simultaneously. To achieve this, every node periodically broadcasts its schedule in a SYNC packet, so that neighbors who hear the SYNC packet start following the same schedule. However, some nodes may not hear the SYNC packets from their neighbors because they have already been running different schedules. Hence, every node must keep awake for a whole SYNC packet interval once in a while, so that different schedules of its neighbors can be heard. A node that has a different schedule from its neighbors may follow both schedules at the same time.

During the awake time, SMAC is similar to IEEE 802.11 [70], which (1) uses RTS/CTS to solve the hidden terminal problem, (2) uses physical carrier sensing and virtual carrier sensing to avoid collision, and (3) uses RTS/CTS/DATA/ACK sequences to guarantee successful unicast transmissions. If a node fails to access the media, it goes back to sleep until the next awake period. If, on the other hand, a node successfully accesses the media, it does not sleep until it finishes the current transmission.

3.1.3 Sleeping at Both Routing and MAC Layers

To employ sleeping Directed Diffusion and SMAC simultaneously, we can either simply implement them as they are, at the routing layer and the MAC layer, respectively, or do cross-layer coordination between the routing layer and the MAC layer to improve the overall performance.

There are various ways to coordinate the routing and MAC layer sleeping. We implement two methods in our simulations. The first cross-layer coordination is based on priority. As sleeping Directed Diffusion puts all the nodes that are not involved in the data delivery to sleep during successive floodings, there is no need to make those nodes wake up at the MAC layer according to its duty cycle, as no data needs to be transmitted. Hence, sleeping Directed Diffusion should have higher priority than SMAC to schedule the nodes. In other words, SMAC only effectively schedules a node when sleeping Directed Diffusion needs to keep this node active. The second cross-layer coordination is differentiation between routing layer sleeping and energy depletion. As SMAC updates a neighbor list at each node by recognizing SYNC packets sent by its neighbors for a given period of time, long sleeping time of a node scheduled by sleeping Directed Diffusion may make the node's neighbors mistakenly drop its information from their neighbor lists, as no SYNC packet is sent from this node during its sleeping time. We, therefore, make a SYNC packet bear the remaining energy of its sender, so that the receiving nodes can easily tell the status of the sender, and remove the sender from its neighbor list only when it is running out of energy.

3.2 Simulations and Discussions

In this section, we analyze the pros and cons of making sensors sleep at individual layers under different network scenarios and application requirements. We also give some preliminary results on the necessity of cross-layer coordination when sensors sleep at both layers. Two metrics are considered. The first metric is the total number of packets received by the sink node. In general, the higher total number of packets, the more information is collected at the sink, which corresponds to a longer application lifetime. However, a high total number of packets received by the sink node does not necessarily imply a good data delivery ratio. Therefore, we define the second metric, data delivery ratio, as the total number of packets actually received by the sink node divided by the number of packets the sink node should ideally receive. Ideally, the sink node receives as many packets as generated by a single reinforced source node during the whole data delivery period. The higher the data delivery ratio is, the fewer packets are lost. In general, for a given application and network deployment, we desire both the total number of packets received by the sink node and the data delivery ratio to be high.

We use ns-2.32 [71] to simulate the performance of all combinations of a non-sleeping/sleeping routing protocol (Directed Diffusion/sleeping Directed Diffusion) and a non-sleeping/sleeping MAC protocol(IEEE 802.11/SMAC[1]). We assume that (1) sensors are static and randomly distributed in a given area, (2) there is only one sink node and more than one source node, (3) the sink node has infinite power supply, while other nodes have 22 J initial energy, (4) each node's power consumptions in transmitting, receiving, and idle status are set the same at 50 mW, based on measurements of CC1000, a radio chip for MICA2 motes [72] and CC2420, a radio chip for IEEE 802.15.4 [25], (5) the size of an application layer data packet is 64 bytes, (6) MAC layer bandwidth is 2Mbps, and (7) the communication range of a sensor is 250 m.

For Directed Diffusion, we assume that the sink node floods an INTEREST packet every 30 s. Each gradient entry in the gradient table is valid for 50 s. The exploratory rate is 1 packet per 100 s. Negative reinforcement check is executed every 6 high rate DATA packet intervals. For sleeping Directed Diffusion, either timer has a 6s time gap. For SMAC, the size of a DATA packet is 50 bytes, the size of a SYNC packet is 9 bytes, and the size of other control packets, like RTS, CTS, and ACK, are 10 bytes. A SYNC packet is sent by each node every 10 duty cycles.

We first look into the performance of individual layer sleeping schemes, namely sleeping Directed Diffusion and SMAC, and then compare their performances under different situations. Finally, we show the performance differences of sleeping at both routing and MAC layers, with and without cross-layer coordination. By default, 30 nodes are randomly distributed in an 800 m $\times$ 800 m area. There is 1 sink node and 5 source nodes. A reinforced source node generates 3 application layer packets per 10 s. For each scenario, 10 topologies are generated, and the results are averaged over 20 simulations, except as noted.

[1]Adaptive listening [13] is used in the simulation of SMAC.

3.2.1 Performance of Single Layer Sleeping

Fig. 3.2 shows the total number of packets received by the sink node over time using different single layer sleeping schemes in one simulation. IEEE 802.11 is used as the common MAC protocol in the simulation of sleeping Directed Diffusion, while Directed Diffusion is used as the common routing protocol in the simulation of SMAC. As we can see, both sleeping Directed Diffusion and SMAC can significantly improve the data delivery period, and hence the total number of packets received by the sink node. However, sleeping Directed Diffusion has a "QoS pause" between 740s to 800s, as the total number of packet received by the sink node remains the same during this period of time. Since a new path reinforcement is always triggered by an exploratory DATA flooding, there might be a gap during which the old path has died but a new path has not been established, and therefore no DATA packets are delivered to the sink. Directed Diffusion does not have a QoS pause because all the nodes in the network die at the same time (a sensor's power consumption in transmitting, receiving and idling are the same), thus no redundant routing paths or source nodes are available to use. As a result, Directed Diffusion leads to a short data delivery period. Moreover, compared to ideal receiving, sleeping Directed Diffusion has almost the same increasing slope in the total number of packets received by the sink node except QoS pauses, but SMAC always has lower total number of packets received by the sink node, because SMAC suffers from more contention and hence more collisions than IEEE 802.11.

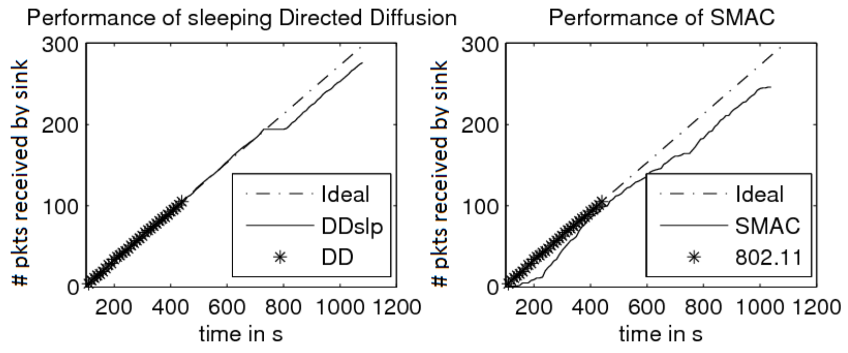


Figure 3.2: Performance of sleeping Directed Diffusion and SMAC.

3.2.2 Performance Comparisons of Individual Layer Sleeping

In this section, we examine the performance of sleeping at individual layers. For simplicity, we mention the combination of Directed Diffusion and IEEE 802.11 as DD802, the combination of Directed Diffusion and SMAC as DDSMAC followed by a specific duty cycle, and the combination of sleeping Directed Diffusion and IEEE 802.11 as DDslp802.

3.2.2.1 Varying Node Density

In this experiment, we vary the node density. Fig. 3.3 shows the performance rendered by 7 schemes under the deployment of 10 nodes, 20 nodes, 30 nodes, 40 nodes and 50 nodes, within the fixed area. As we can see, without sleeping, DD802 performs the same in both total number of packets received by the sink node and the data delivery ratio, no matter what the node density is. This provides a baseline to judge the performance of all the sleeping schemes. Specifically, DD802 on average has 105 packets received by the sink node in total with 100% data delivery ratio. However, its data delivery period is short in the absence of any sleeping technique.

DDSMAC, on the other hand, shows a diversity of performance under different duty cycles and different node densities. For a given node density, generally, the lower the duty cycle is, the longer alive time every sensor in the network can have, which is beneficial to receiving more packets at the sink node. However, a low duty cycle also leads to severe contention and consequently higher probability of collision. One one hand, collisions may affect the SYNC packet exchange, so that some sensors cannot communicate with each other, and hence have to drop packets. On the other hand, consistent collisions may either lead to packet drop once the packet is retransmitted up to the limit, or incur long packet delay, so that negative reinforcement may be initiated by Directed Diffusion, and then the data delivery path is cut off. Therefore, lower duty cycles always have lower data delivery ratio, but lower duty cycles may not necessarily mean a high total number of packets received by the sink node. As shown in Fig. 3.3, the best duty cycle for DDSMAC in terms of total number of packets received by the sink node varies according to the node density, which reflects the contention in the network. As the node density increases, the duty cycle with highest total number of packets received by the sink node increases from 10% for 10 nodes to 40% for 50 nodes. Note that when the node density is very high, more than 40 nodes in our experiment, DDSMAC under all duty cycles performs worse than DD802 in both total number of packets received by the sink node and the data delivery ratio. Therefore, it is not worth sleeping only at the MAC layer in this case. For a given duty cycle, both the total number of packets received by the sink node and the data delivery ratio decrease as the node density increases. This can be explained by the fact that high node density causes high contention.

On the contrary, DDslp802 has higher total number of packets received by the sink node as the node density increases. Higher node density implies more routing redundancy. DDSMAC wastes the routing redundancy, but DDslp802 takes advantage of this added redundancy. Specifically, DDSMAC wakes up redundant routing nodes the same as it wakes up reinforced routing nodes. Hence when a reinforced routing path runs out of energy, all the other redundant paths run out of energy as well. Hence for DDSMAC, the maximum total number of packets received by the sink node is upper-bounded by the the total number of packets that can be generated by a single source node. DDslp802, however, saves the energy of those redundant paths by allowing redundant routing nodes to sleep, so that when one path dies, another path can be used to deliver packets. The higher the node density is, the more redundant paths will be available to improve the data delivery period as well as the total number of packets received by the sink node. However, DDslp802 suffers from QoS pauses. The more often it changes to a new routing path, the more chances it introduces a period of QoS pause. Hence, DDslp802 has worse data delivery ratio as the node density increases. However, the data delivery ratio of DDslp802 decreases much slower than the data delivery ratio of DDSMAC. Fig. 3.3 also shows that QoS pause impairs the data delivery ratio more than SMAC unreliability at low node density, but it impairs the data delivery ratio less than SMAC unreliability at high node density.

Due to space limitations, the standard deviations of the total number of packets received by the sink node and the data delivery ratio are not shown in the figures. However, we observed that DD802 and DDslp802 have very small standard deviations for both the total number of packets received by the sink node and the data delivery ratio, while DDSMAC has larger standard deviations as the node density increases or as

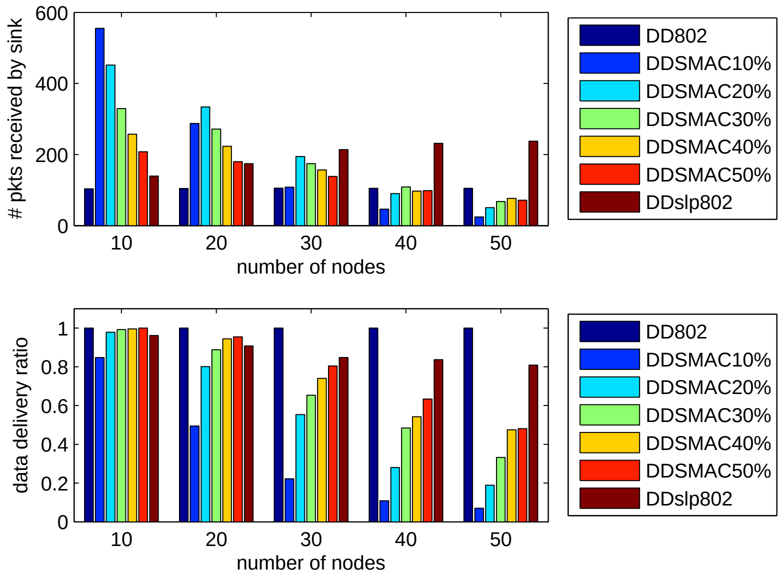


Figure 3.3: Performance comparison under varying node density.

the duty cycle decreases. Similar results are observed in all the experiments throughout the chapter.

3.2.2.2 Varying Network Scale

In this experiment, we fix the node density, but vary the network scale. 17, 23, 30, 38 and 47 nodes are placed in a 600 m $\times$ 600 m, 700 m $\times$ 700 m, 800 m $\times$ 800 m, 900 m $\times$ 900 m and 1000 m $\times$ 1000 m area, respectively.

Fig. 3.4 shows the performance of the 7 schemes under different network scales with the same node density. DD802 performs almost the same as in the experiment of varying node density in a fixed area, with an average of 105 packets received by the sink node in total and 100% data delivery ratio. Obviously, neither node density nor network scale influences the performance of DD802.

DDSMAC has an overall decreasing performance as the network scale increases. Since a large network scale implies that source nodes are on average more hops away from the sink node, packets from a source node may experience worse synchronization, longer delay and higher dropping probability on the routing path. Therefore, as the network scale increases, the best duty cycle for DDSMAC in terms of total number of

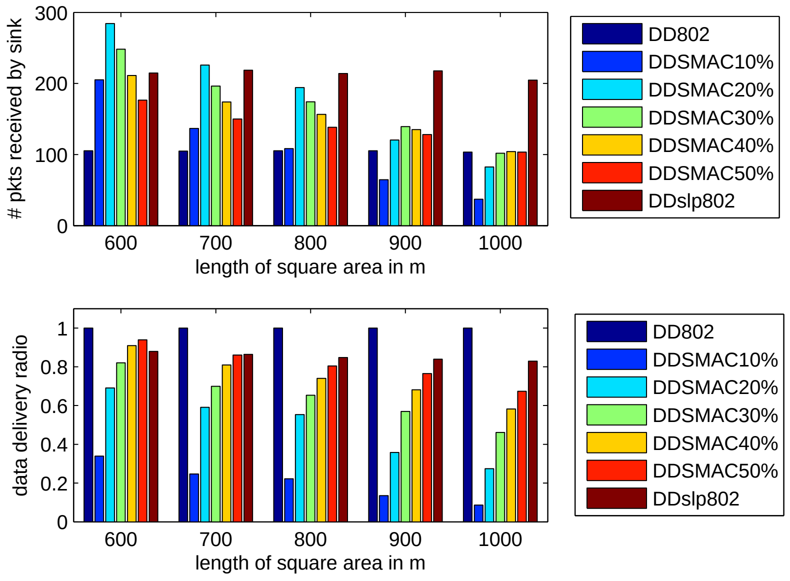


Figure 3.4: Performance comparison under varying network scale.

packets received by the sink node increases from 20% for 600 m $\times$ 600 m area to 40% for 1000 m $\times$ 1000 m area, due to the fact that larger duty cycles always have better data delivery ratios.

DDslp802, on the other hand, has a steady performance in both total number of packets received by the sink node and the data delivery ratio. As fixed node density guarantees that a sensor has on average a fixed number of neighbors, enlarging the network scale does not increase the routing redundancy. Meanwhile, IEEE 802.11 has reliable data delivery, hence DDslp802 does not suffer from extended multihop transmission.

3.2.2.3 Varying Number of Sources

In this experiment, the number of source nodes varies from 2 to 26. Fig. 3.5 shows the performance of the 7 schemes. DD802, as usual, has constant performance.

Overall, DDSMAC has an obvious improvement in the total number of packets received by the sink node when the number of source nodes increases from 2 to 5, but the improvement slows down when the number of source nodes continues to increase. As the number of source nodes increases, the sink node can on average reach one of

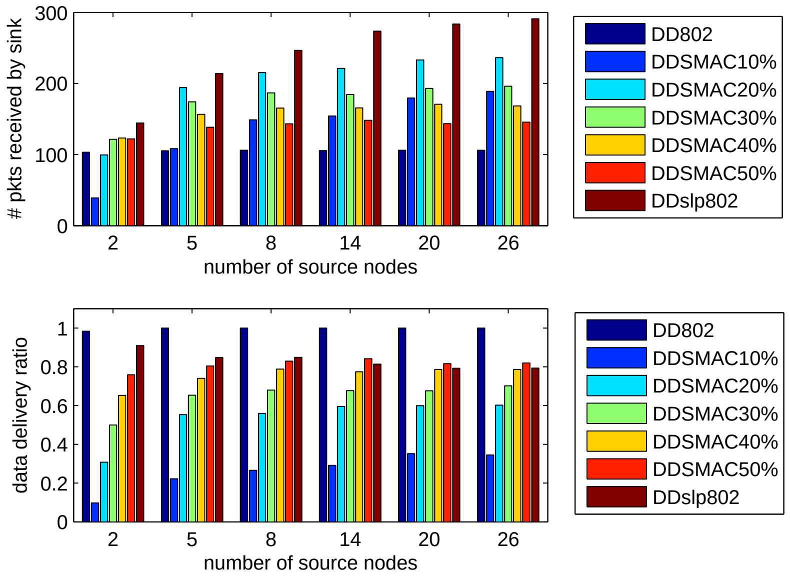


Figure 3.5: Performance comparison under varying number of source nodes.

the source nodes in fewer hops. The fewer hops a transmission experiences, the higher data delivery ratio it will have, see Fig. 3.4. However, as the number of source nodes increases, the probability that the sink node has at least one neighboring source node improves from fast to slowly (more than one 1-hop source node does not help DDSMAC since DDSMAC does not utilize routing redundancy). Accordingly, the improvement in the total number of packets received by the sink node slows down.

Compared with DDSMAC, which improves in both the total number of packets received by the sink node and the data delivery ratio with increasing numbers of source nodes, DDslp802 increases in the total number of packets received by the sink node, but decreases in data delivery ratio. This is because source nodes are the second redundancy that sleeping Directed Diffusion could use besides redundant routing nodes. Since Directed Diffusion reinforces one path (one source node) at once to deliver data, other redundant source nodes are put to sleep for later use. Hence, once a source node dies, another source node can be reinforced to continue the data delivery. However, the more source nodes to reinforce, the more chances DDslp802 will incur a QoS pause. Consequently, the data delivery ratio decreases slightly as the number of source nodes increases.

3.2.2.4 Varying Data Rate

In this experiment, the application data rate at each reinforced source node varies from 1 packet per 10 s to 8 packets per 10 s. Fig. 3.6 shows the total number of packets received by the sink node of the 7 schemes scaled by the application data rate and their corresponding data delivery ratio. The total number of packets received by the sink node of DD802 increases proportionally to the application data rate, while the data delivery ratio is 100%.

In absolute terms, DDSMAC also receives more packets as the application data rate increases, but relatively, the total number of packets received by the sink node of DDSMAC is decreasing if scaled by the application data rate. The decreasing performance of DDSMAC can also be seen in its data delivery ratio. This is because SMAC cannot provide reliable data delivery, as shown in section 3.2.1, due to the contention in the network. Higher application data rates mean more packets to send, and hence more contention and packet loss in a fixed period of time. As the application data rate increases, the best duty cycle in terms of the total number of packets received by the sink node increases from 20% for 1 packet per 10 s to 30% for 8 packets per 10 s. Larger duty cycles can alleviate contention in the network, as sensors are sleeping for a shorter time.

DDslp802, on the other hand, has the total number of packets received by the sink node proportional to the application data rate and maintains the same data delivery ratio as the application data rate changes. As IEEE 802.11 provides reliable data delivery, the data delivery ratio of DDslp802 drops only because of the QoS pauses introduced by sleeping Directed Diffusion. When the node density and the number of sources are kept the same, the redundancy that DDslp802 can utilize does not change. Hence, DDslp802 reinforces on average the same number of redundant paths, and consequently has the same chances of introducing a QoS pause.

3.2.3 Performance of Sleeping at Both Layers

In this experiment, we compare the performance of sleeping schemes at both routing and MAC layers, with and without cross-layer coordination, as described in section 3.1.3. For simplicity, we mention the combination of sleeping Directed Diffusion and

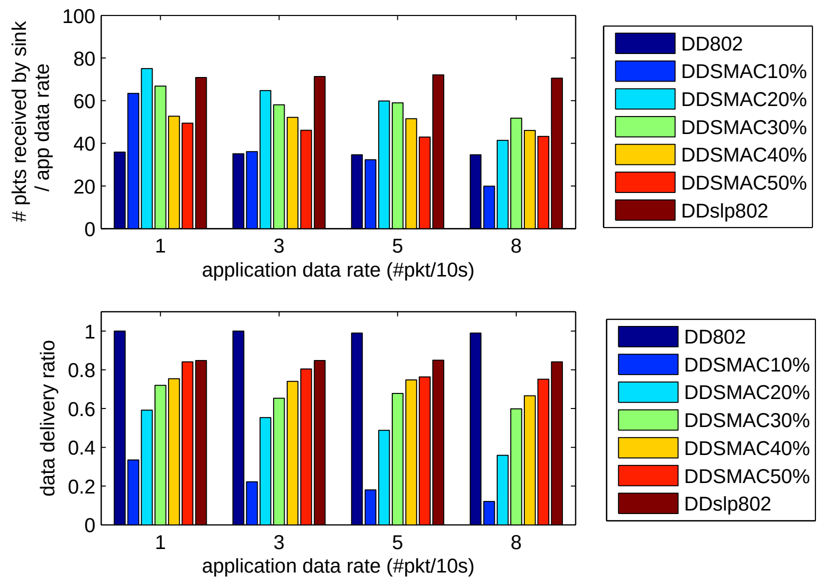


Figure 3.6: Performance comparison under varying application data rate.

SMAC with its best duty cycle in terms of the total number of packets received by the sink node under DDSMAC as DDslpSMAC, followed by -NC or -C for the case of without coordination or with coordination, respectively.

3.2.3.1 Varying Node Density

Fig. 3.7 shows the total number of packets received by the sink node and the data delivery ratio of 5 schemes as the node density varies from 10 nodes to 50 nodes randomly placed in a given 800mX800m area. As we can see, DDslpSMAC-NC has very similar performance in both total number of packets received by the sink node and the data delivery ratio as DDSMAC even though sensors can now sleep at both layers. Since no coordination is implemented between the routing layer and the MAC layer, sleeping Directed Diffusion does not have higher priority than SMAC to schedule the sensors. When a sensor is turned off by sleeping Directed Diffusion, it will still be woken up periodically by SMAC according to its duty cycle. Hence, sensors are mostly following SMAC to sleep and wake up.

On the other hand, with certain coordination, DDslpSMAC-C significantly improves the total number of packets received by the sink node compared with DDSMAC, since it not only reduces idle listening during data delivery but also utilizes network redundancy. When the node density is not very high, DDslpSMAC-C performs the best in terms of the total number of packets received by the sink node among all the sleeping schemes. However, when the node density increases, DDslpSMAC-C cannot achieve as high total number of packets received by the sink node as DDslp802, since the severe contention at the MAC layer greatly impairs the most vulnerable but most important data delivery at the routing layer - unicasting positive reinforcement packets and unicasting high rate DATA packets. The loss of positive reinforcement packets delays the path establishment, while the loss of DATA packets impairs the total number of packets received by the sink node directly. Hence, DDslpSMAC-C cannot further improve the the total number of packets received by the sink node compared with DDslp802, although sensors sleep at both layers. DDslpSMAC-C has a data delivery ratio lower than the data delivery ratio of either DDSMAC or DDslp802, because both QoS pauses and SMAC unreliability are degrading the performance. Meanwhile, DDslpSMAC-C has similar standard deviations with DDSMAC in either the total number of packets received by the sink node or the data delivery ratio, while DDslp802 has smaller standard deviations (not shown in the figure).

3.2.3.2 Varying Network Scale

Fig. 3.8 shows the total number of packets received by the sink node and the data delivery ratio of 5 schemes as the network scale varies. Specifically, remaining the same node density, 17, 23, 30, 38, and 47 nodes are placed in a 600 m $\times$ 600 m, 700 m $\times$ 700 m, 800 m $\times$ 800 m, 900 m $\times$ 900 m, and 1000 m $\times$ 1000 m area, respectively. Again, DDslpSMAC-NC performances similarly with DDSMAC, since there is no coordination between the routing layer and the MAC layer to give sleeping Directed Diffusion a higher priority to put sensors to sleep.

DDslpSMAC-C, however, outperforms all other sleeping schemes when the network scale is small. This is because DDslpSMAC-C can take advantage of both sleeping Directed Diffusion and SMAC after proper coordination. However, as the network scale increases, SMAC suffers from poor synchronization and extended multihop transmissions, and hence more positive reinforcement packets and unicasting high rate DATA packets may be dropped. As a result, DDslpSMAC-C has lower total number

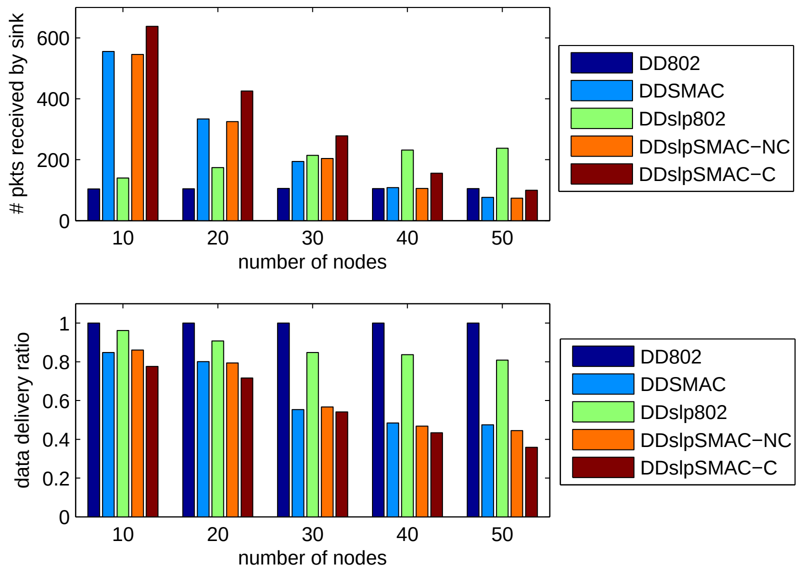


Figure 3.7: Performance of multi-layer sleeping with/without cross-layer coordination under varying node density.

of packets received by the sink node when the network scale is large, while DDslp802 performs the best. Since IEEE 802.11 has reliable data delivery, DDslp802 is not affected by the extended multihop transmissions. The performance of DDslp802 remains the same as the network scale varies.

3.2.3.3 Varying Number of Sources

Fig. 3.9 shows the total number of packets received by the sink node and the data delivery ratio of 5 schemes as the number of source nodes varies from 2 to 26. Again, DDslpSMAC-NC performances similarly with DDSMAC.

DDslpSMAC-C, however, outperforms all other sleeping schemes when the number of source nodes increases. This is because (1) sleeping Directed Diffusion can utilize the increasing source node redundancy to extend the network lifetime, and (2) SMAC can provide good data delivery ratio (in the scenario of 30 nodes in a 800 m $\times$ 800 m area with 3 application layer packets per 10 s at each reinforced node) to support routing signaling and date transmission. However, when the number of source nodes is very small, e.g., 2 source nodes, DDslp802 is a better choice than DDslpSMAC-C.

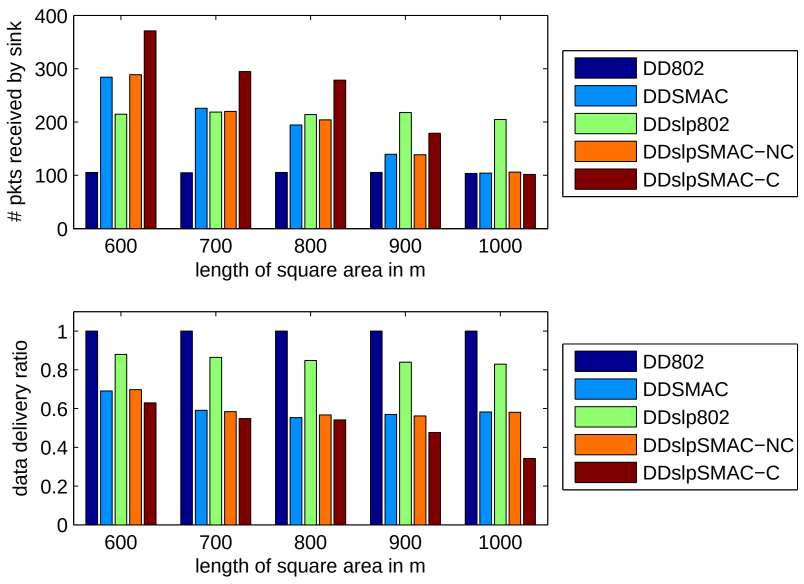
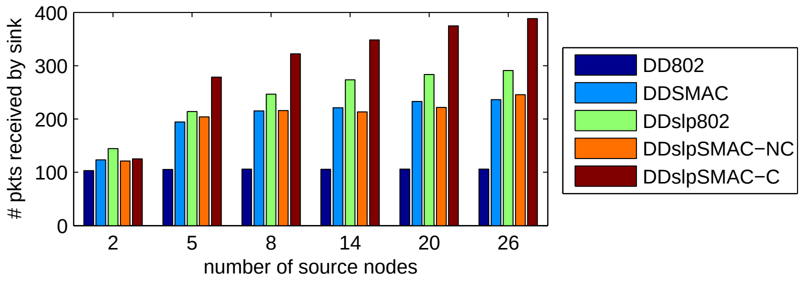


Figure 3.8: Performance of multi-layer sleeping with/without cross-layer coordination under varying network scale.

This is because SMAC on one hand extends the network lifetime, and on the other hand lowers the data delivery ratio and impairs the path reinforcement in the sleeping Directed Diffusion. Hence, when the network redundancy is low, DDslpSMAC-C may not earn more packet receptions from sleeping Directed Diffusion, but loses more packet receptions due to unreliable SMAC. In this case, IEEE 802.11 guarantees that network redundancy can be utilized by sleeping Directed Diffusion, and therefore, has the best performance in terms of the total number of packets received by the sink node.

3.2.3.4 Varying Data Rate

In this experiment, the application data rate at each reinforced source node varies from 1 packet per 10 s to 8 packets per 10 s. Fig. 3.10 shows the total number of packets received by the sink node of the 5 schemes scaled by the application data rate and their corresponding data delivery ratio. Again, DDslp802 and DDslpSMAC-NC has similar performance. But DDslpSMAC-C performs the best in terms of the total number of packets received by the sink node when the application data rate is not very high. When the application data rate increases to 8 packets per 10 s, DDslp802 outperforms



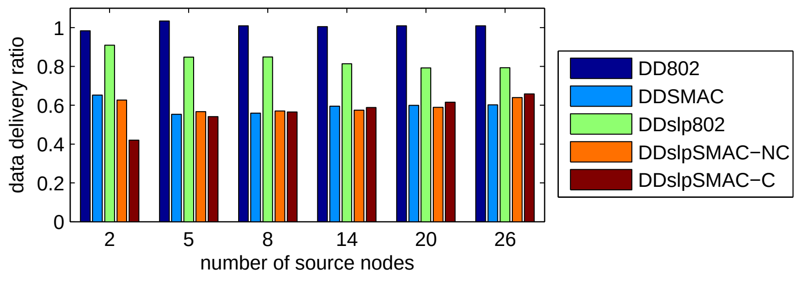


Figure 3.9: Performance of multi-layer sleeping with/without cross-layer coordination under varying number of sources.

DDslpSMAC-C, since higher data rate means higher contentions in the network, and hence poorer SMAC reliability in data delivery. Therefore, in the high contention scenarios, DDslp802 is a better choice over DDslpSMAC-C in terms of the total number of packets received by the sink node, as MAC layer reliability is necessary for sleeping at Directed Diffusion to take effect.

The above results show the condition and benefit of employing cross-layer coordination between the routing and MAC layers to achieve higher total number of packets received by the sink node under various network scenarios and application requirements. Specifically, sleeping at both routing and the MAC layers with proper coordination can achieve better performance in terms of the total number of packets received by the sink node when SMAC can provide good reliability in data delivery. The benefit of sleeping at both layers is obvious when the contention in the network is low, network scale is small, or the network redundancy is high. Hence, a smart decision can be made by the power management of wireless sensor networks to dynamically choose the best sleeping scheme (single layer sleeping or multi-layer sleeping with coordination) as the network condition changes over time. Moreover, we believe the performance

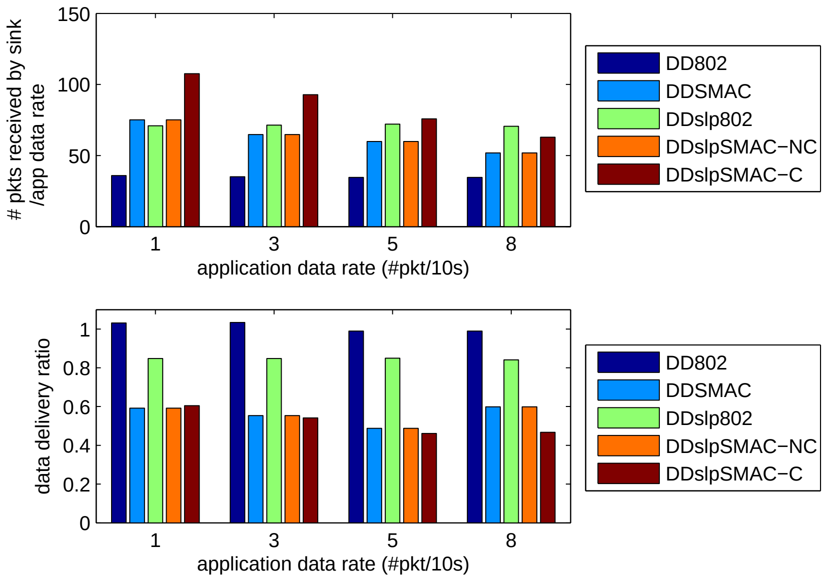


Figure 3.10: Performance of multi-layer sleeping with/without cross-layer coordination under vary data rate.

of DDslpSMAC-C can be further improved and may outperform single layer sleeping schemes all the time as more sophisticated cross-layer coordination is employed.

3.3 Summary

In this chapter, we analyze sleeping schemes conducted by routing protocols and MAC protocols individually and simultaneously, for wireless sensor networks, in order to determine the best method for sleeping under different node densities, network scales, numbers of source nodes and application data rates. We also investigate various MAC layer duty cycles in the performance comparisons. While conclusions are made by simulating networks that run routing protocols with/without sleeping (Directed Diffusion and our newly proposed sleeping Directed Diffusion) and MAC protocols with/without sleeping (IEEE 802.11 and SMAC), the conclusions can also be applied to other routing protocols that turn off sensors when they are not involved in the data delivery, and other MAC protocols that use duty cycling to save energy.

In general, our results show that routing layer sleeping is more suitable for networks with high redundancy or high contention, while MAC layer sleeping is more sensitive to contention, and hence is a good choice for light traffic applications under small scale and sparse networks. Furthermore, we show that cross-layer coordination can significantly improve the total number of packets received by the sink node under low contention scenarios, when routing layer sleeping and MAC layer sleeping are employed simultaneously. Therefore, a smart decision can be made by an adaptive sensor sleeping solution that dynamically switches to a sleeping scheme at the routing layer or the MAC layer or both layers with cross-layer coordination, as the network conditions or application requirements change over time. Moreover, more sophisticated cross-layer coordination has the potential to further improve the total number of packets received by the sink node and outperform single layer sleeping schemes in all cases.

However, in order to implement adaptive sensor sleeping solutions that adjust to dynamic network conditions, it is important to have a full understanding of how the protocols operate under different network conditions. In the next few chapters, we look specifically at the performance of duty-cycled MAC protocols, developing analytical models to determine the throughput, latency and energy performance of both synchronous and asynchronous duty-cycled MAC protocols.

Chapter 4

A General Method to Analyze the Performance of Duty-cycled MAC Protocols

In Chapter 3, our simulation results show promising gains in making sensors sleep smartly by adding coordination between different layers. However, to provide an adaptive sensor sleeping solution that can dynamically decide the sleep strategy of a wireless sensor network according to the network conditions and application requirements, it is necessary to estimate the performance of the protocol at every single layer, including the MAC layer. However, there are many different MAC protocols that can be used in wireless sensor networks, hence it is desirable to have a general model that can handle different MAC protocols with protocol-specific settings.

Analytical models have been proposed to evaluate the performance of a specific MAC protocol, such as IEEE 802.11 [41], IEEE 802.15.4 [42], S-MAC [28][43], and OMAC [37]. However, these models are fundamentally different and cannot be generalized, as different protocols have different media access rules. Hence, the application of these models is limited to the particular protocols they model. Some other work focused on analyzing a specific performance metric, for example, delay, or energy consumption, for a specific MAC protocol or a series of MAC protocols that have the same characteristics. However, their performance evaluations are constrained for only one metric of a protocol's performance, and hence, their models cannot be used to determine the trade-offs in different performance metrics. Meanwhile, so far as we know, there is no work that can generalize both synchronized MAC protocols and asynchronous MAC protocols.

We therefore propose a general methodology to analyze the throughput, delay, and energy consumption of duty-cycled MAC protocols (both synchronized and asynchronous) [27]. To handle different MAC protocols, our approach decouples the performance analysis problem into two parts. Specifically, we propose a Markov model to describe the queueing behavior of duty-cycled nodes with finite queue capacity. Based on the Markov model, the stationary probability for each node to have an empty queue π_{EmptyQ} can be obtained as a function of the probability for each node to transmit a data packet p in a cycle, i.e., $\pi_{EmptyQ} = f(p)$. On the other hand, we propose a general methodology to handle the protocol-specific media access rules, which can determine the probability for each node to transmit a data packet in a cycle p as a function of the probability of having an empty queue π_{EmptyQ} at each node, i.e., $p = g(\pi_{EmptyQ})$. Therefore, solving these two functions, the value of π_{EmptyQ} and p that the investigated protocol is operating on can be obtained. Once we know π_{EmptyQ} and p, the throughput, delay and energy consumption of the network can be determined.

Although the method to obtain $p = g(\pi_{EmptyQ})$ is protocol-specific, as different MAC protocols have different media access rules, our proposed methodology to analyze the performance of a duty-cycled MAC protocol using the Markov model is general. In this dissertation we apply our proposed method to two different duty-cycled MAC protocols, one is S-MAC, a synchronized duty-cycled MAC protocol (Chapter 5), and the other is X-MAC, an asynchronous duty-cycled MAC protocol (Chapter 6). We show that the analytical throughput, delay and energy consumption using our model match the simulation results, and we also show how our model can be used to optimize the protocol parameters to achieve a desirable performance.

The rest of this chapter is organized as follows. Section 4.1 presents our proposed Markov models to describe the queueing behavior of a duty-cycled node with and without retranmissions. Section 4.2 provides the general methodology to analyze the network throughput, packet delay, and energy consumption using our proposed Markov models. Section 4.3 summarizes the chapter and introduces our next step to validate the proposed Markov models and performance analysis methodology, which will be used in Chapters 5 and 6 when applying our model to S-MAC and X-MAC, respectively.

4.1 General Markov Models for Queueing Behavior of Duty-cycled Nodes

We propose two Markov models to describe the queueing behavior of duty-cycled nodes with a fixed cycle length. The first Markov model does not support retransmissions, while the second Markov model supports retransmissions. Our Markov models assume that (1) packets arrive at each node independently, (2) each node can buffer (a finite number of) DATA packets in a FIFO queue, (3) the channel is ideal (no fading and no capture effect), (4) there is only one transmission opportunity and one DATA packet reception per node per cycle, and (6) every node has a constant probability of transmitting a DATA packet in a cycle regardless of any node's queue length (similar assumptions were made in [41][50], and were verified as good approximations of real scenarios). Table 4.1 lists some of the notations that are used throughout this chapter and the following two chapters.

4.1.1 Duty-cycled Nodes Without Retransmissions

The proposed Markov model without retransmission support has a finite number of states, each of which represents a different status of a node, i.e., a different queue length, at the wake-up instant of a cycle. A node may change status cycle by cycle, corresponding to the transition from one state to another in the Markov model. Fig. 4.1 shows the proposed Markov model with a queue capacity Q. This Markov model has $Q + 1$ states, each of which, from left to right, corresponds to 0 packets in the queue, 1 packet in the queue, to Q packets in the queue (full queue). Specifically, when the queue is not empty, a node will attempt to access the media to transmit a DATA packet. A DATA packet is removed from the queue either when it is transmitted successfully, or when it encounters a collision as no retransmission is allowed. Meanwhile, a DATA packet is dropped when the queue overflows. Hence, the transition probabilities from one state to another can be described as follows.

Table 4.1: Notations

Symbol	Quantity
N	number of nodes in the network
Q	queue capacity in units of a DATA packet
R	retransmission limit
d	duty cycle
T	length of a cycle
W	contention window size in units of a time slot
λ	DATA packet arrival rate at the MAC layer
A_k	probability of k packets arriving in a cycle $A_k = e^{-\lambda T}(\lambda T)^k/k!$
$A_{\geq}k$	probability of no less than k DATA packets arriving in a cycle $A_{\geq}k = 1 - \sum_{i=0}^{k-1} A_i$
p_s	probability of successfully transmitting a DATA packet
p_f	probability of transmission failure of a DATA packet
p	probability of winning the contention $p = p_s + p_f$
S	MAC layer DATA packet size
THR	network throughput
D_C	packet contending delay
D_Q	packet queuing delay
D	packet delay $D = D_C + D_Q$
E	energy consumption of a node in a cycle
P	average energy consumption per second
L	lifetime of each node in the network

$$P_{0,i} = A_i, i = 0..Q-1 \tag{4.1}$$

$$P_{0,Q} = A_{\geq Q} \tag{4.2}$$

$$P_{i,j-1} = p \cdot A_0, i = 1..Q \tag{4.3}$$

$$P_{i,j} = p \cdot A_{j-i+1} + (1-p) \cdot A_{j-i}, i = 1..Q-1, j = i..Q-1 \tag{4.4}$$

$$P_{i,Q} = p \cdot A_{\geq Q-i+1} + (1-p) \cdot A_{\geq Q-i}, i = 1..Q \tag{4.5}$$

$$P_{i,j} = 0, i = 2..Q, j = 0..i-2 \tag{4.6}$$

Equations (4.1) and (4.2) describe the fact that all the transitions from the empty-queue state to a non-empty-queue state depend only on new packet arrivals. Equations (4.3) and (4.6) describe the fact that a node can only transmit one DATA packet per cycle with a probability p, and the probability of having one packet less in the queue equals the probability of winning the contention times the probability of no packet arrivals in a cycle. Moreover, (4.4) and (4.5) describe the fact that the probability of having a non-decreasing queue can be divided into two parts depending on whether the oldest DATA packet in the queue wins the contention (first term) or not (second term). Finally, (4.2) and (4.5) show that packets are dropped when the queue overflows.

The proposed Markov model with state space $\tilde{S} = 0, 1, ..., Q$ and transition matrix $\tilde{P}$ has a unique stationary distribution $\pi = (\pi_0, .., \pi_Q)$ since the Markov model is irreducible and aperiodic. Therefore, $\pi_i \geq 0$ for any $s_i \in \tilde{S}$,

$$\sum_{s_i \in \tilde{S}} \pi_i = 1, \pi\tilde{P} = \pi \tag{4.7}$$

Assuming packet arrival information (λ, A_k, and $A_{\geq}k$) is known, the probability of winning the contention p becomes the only variable in the transition matrix $\tilde{P}$. Since π

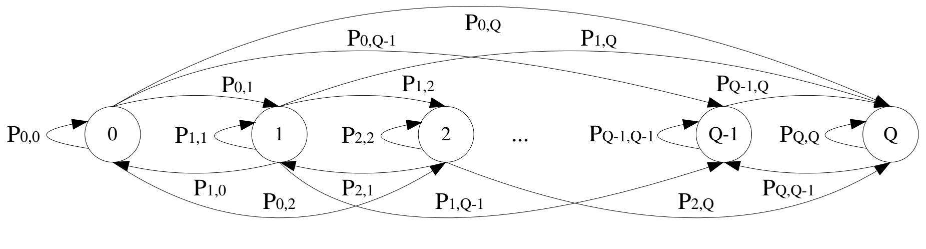


Figure 4.1: Markov model for queueing behavior of duty-cycled nodes without retransmissions.

is the unique solution for (4.7), for any $s_i \in \tilde{S}$, π_i can be represented as a function of p. Specifically, let function $f(\cdot)$ describe the relationship between π_0 and p, i.e.,

$$\pi_0 = f(p) \tag{4.8}$$

According to equation (4.8), for a given probability for each node to win the contention, p, the stationary probability of the empty-queue state π_0 can be obtained using the proposed Markov model.

We propose this Markov model instead of using an M/M/1/Q queuing model because (1) the data packet arrivals at each node in our Markov model can be from distributions other than Poisson, and (2) the service rate at each node (the packet transmission rate at each node in a cycle) is unknown, and it depends on the contention in the network.

4.1.2 Duty-cycled Nodes With Retransmissions

For duty-cycled MAC protocols with retransmissions, the retransmission stage as well as the queue length determines the status of a node. Specifically, when R retransmissions are supported, there are $R + 1$ retransmission stages to describe the retransmission status. Note that a node with an empty queue is always in retransmission stage 0. Fig. 4.2 shows the retransmission Markov model for duty-cycled MAC protocols with R retransmissions and queue capacity Q. It has $Q \cdot (R + 1) + 1$ states, each of which is represented by two indices: retransmission stage and queue length.

We first look at the transitions from the empty-queue state. Their probabilities depend only on the new packet arrivals.

$$P_{(0,0)->(0,i)} = A_i, i = 0..Q - 1 \tag{4.9}$$

$$P_{(0,0)->(0,Q)} = A_{\geq Q} \tag{4.10}$$

Then, we consider the transitions within one retransmission stage. Equations (4.11), (4.12) and (4.13) describe the transitions within retransmission stage 0, whereas (4.14)

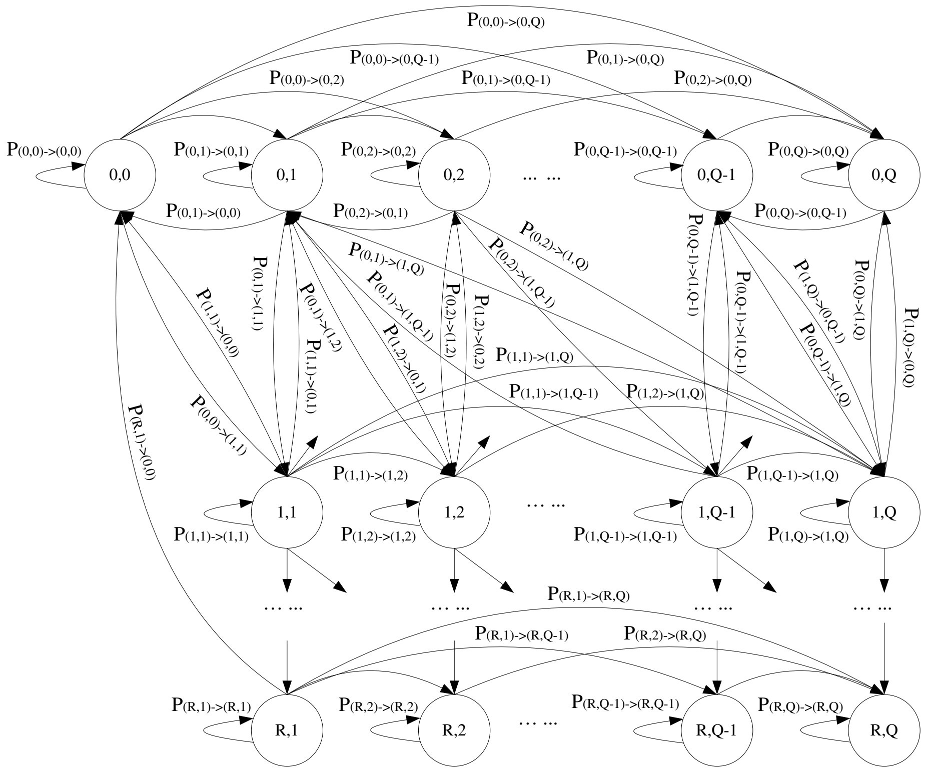


Figure 4.2: Markov model for queueing behavior of duty-cycled nodes with retransmissions.

and (4.15) describe the transitions within a non-zero retransmission stage.

$$P_{(0,j)->(0,j-1)} = p_s \cdot A_0, j = 1..Q \tag{4.11}$$

$$P_{(0,j)->(0,k)} = p_s \cdot A_{k-j+1} + (1-p) \cdot A_{k-j}, j = 1..Q-1, k = j..Q-1 \tag{4.12}$$

$$P_{(0,j)->(0,Q)} = p_s \cdot A_{\geq Q-j+1} + (1-p) \cdot A_{\geq Q-j}, j = 1..Q \tag{4.13}$$

$$P_{(i,j)->(i,k)} = (1-p) \cdot A_{k-j}, i = 1..R, j = 1..Q-1, k = j..Q-1 \tag{4.14}$$

$$P_{(i,j)->(i,Q)} = (1-p) \cdot A_{\geq Q-j}, j = 1..Q \tag{4.15}$$

Next, we examine the transitions from one retransmission stage to the adjacent higher retransmission stage. These transitions correspond to the event that a node experiences an RTS collision.

$$P_{(i,j)->(i+1,k)} = p_f \cdot A_{k-j}, i = 0..R-1, j = 1..Q-1, k = j..Q-1 \tag{4.16}$$

$$P_{(i,j)->(i+1,Q)} = p_f \cdot A_{\geq Q-j}, i = 0..R-1, j = 1..Q \tag{4.17}$$

Finally, we describe the probabilities of transitions from a non-zero retransmission stage to retransmission stage 0. These transitions correspond to the events either that a retransmitted DATA packet is successfully delivered, described in (4.18) and (4.19), or that a retransmitted DATA packet is discarded due to reaching the retransmission limit, described in (4.20) and (4.21).

$$P_{(i,j)->(0,k)} = p_s \cdot A_{k-j+1}, i = 1..R-1, j = 1..Q-1, k = j-1..Q-1 \tag{4.18}$$

$$P_{(i,j)->(0,Q)} = p_s \cdot A_{\geq Q-j+1}, i = 1..R-1, j = 1..Q \tag{4.19}$$

$$P_{(R,j)->(0,k)} = p \cdot A_{k-j+1}, j = 1..Q-1, k = j-1..Q-1 \tag{4.20}$$

$$P_{(R,j)->(0,Q)} = p \cdot A_{\geq Q-j+1}, j = 1..Q, k = j-1..Q \tag{4.21}$$

Transitions that are not listed above have a probability of 0.

The retransmission Markov model assumes that every node has constant p_s and p_f regardless of any node's queue length or retransmission stage. This assumption is also verified as a good approximation of the real case in [41][49][55] and through our model validation.

The retransmission Markov model with state space $\tilde{S} = \{(0,i) \cup (j,k) | i = 0..Q, j = 1..R, k = 1..Q\}$ and transition matrix $\tilde{P}$ has a unique stationary distribution

$\pi = \{\pi_{(i,j)} | (i,j) \in \tilde{S}\}$ since the extended Markov model is irreducible and aperiodic. Therefore, $\pi_{(i,j)} \geq 0$ for any $(i,j) \in \tilde{S}$,

$$\sum_{(i,j)\in\tilde{S}} \pi_{(i,j)} = 1, \pi\tilde{P} = \pi \tag{4.22}$$

Assuming packet arrival information (λ, A_k, and $A_{\geq}k$) is known, the probability for each node to successfully transmit a DATA packet p_s, and the probability of DATA transmission failure at each node p_f are the only two variables in the transition matrix $\tilde{P}$. Since π is the unique solution for (4.22), for any $(i,j) \in \tilde{S}$, $\pi_{(i,j)}$ can be represented as a function of p_s and p_f. Specifically, let function $\tilde{f}(\cdot)$ describe the relationship between $\pi_{(0,0)}$ and p_s and p_f, i.e.,

$$\pi_{(0,0)} = \tilde{f}(p_s, p_f) \tag{4.23}$$

According to equation (4.23), for a given probability for each node to successfully transmit a DATA packet p_s and a given probability for each node to encounter a DATA transmission failure p_s, the stationary probability of the empty-queue state $\pi_{(0,0)}$ can be obtained using the retransmission Markov model.

4.2 A General Methodology of Protocol-specific Analysis

Our general Markov models holds for any duty-cycled MAC protocol with a fixed cycle length. For the no retransmission Markov mode, it provides a relationship between the stationary probability of the empty-queue state π_0 and the probability for each node to win the contention p in a cycle, as shown in (4.8). However, another relationship between π_0 and p is needed together with (4.8) to solve for both π_0 and p, and finally using these values to obtain the network throughput, delay, and energy consumption. Since (4.8) obtains π_0 as a function of p, here we show how to use protocol-specific media access rules to obtain p as a function of π_0, i.e.,

$$p = g(\pi_0) \tag{4.24}$$

as our second relationship between π_0 and p.

When π_0 is given, every node has a probability of π_0 to have an empty queue. In other words, every node has a probability of $1 - \pi_0$ to have a packet to send in a cycle. For a given protocol, the probability for each node to win the contention p can be obtained with the knowledge of π_0 and the media access rules of the protocol. Since different protocols have different media access rules, (4.24) is protocol-specific. We provide examples of how to obtain (4.24) for both S-MAC and X-MAC in Chapter 5 and Chapter 6.

Solving (4.8) and (4.24), the stationary probability of the empty-queue state π_0 and the probability for each node to win the contention p can be obtained. Meanwhile, plugging p into (4.1)-(4.6), the stationary distribution of the Markov model π can be found. These values, as well as the media access rules of the investigated protocol, enable us to analyze the throughput, delay and energy consumption of the network.

For the retransmission Markov mode, it provides a relationship that the probability for each node to successfully transmit a DATA packet p_s and the probability for each node to encounter a DATA transmission failure p_f are functions of the stationary probability of the empty-queue state $\pi_{(0,0)}$. However, another relationship between $\pi_{(0,0)}$ and p_s, p_f is needed together with (4.23) to solve for both $\pi_{(0,0)}$ and p_s, and finally using these values to obtain the network throughput, delay, and energy consumption.

When $\pi_{(0,0)}$ is known, p_s and p_f can be obtained according to the media access rules of the investigated protocol. Similar to the way we obtain p, every node has a packet to send with a probability of $1 - \pi_{(0,0)}$ in a cycle. For a given $\pi_{(0,0)}$, the probability for a node to successfully transmit a DATA packet, p_s and the probability for each node to have a DATA transmission failure p_f, are determined by the number of nodes in the network N and the manner in which nodes compete with each other. The relationship between p_s, p_f and $\pi_{(0,0)}$ can be described as

$$(p_s, p_f) = g(\pi_{(0,0)}) \tag{4.25}$$

Solving (4.23) and (4.25), $\pi_{(0,0)}$, p_s, and p_f can be obtained. Meanwhile, plugging p_s and $p_f$into (4.9)-(4.21), the stationary distribution of the Markov model π can be found. These values, as well as the media access rules of the investigated protocol, enable us to analyze the throughput, delay and energy consumption of the network.

4.2.1 System Model

We examine the throughput, delay and average energy consumption per second according to the following system model. A certain number of nodes create a connected network. The nodes are homogeneous in initial energy, power and communication capabilities. Every node has a finite queue to buffer the incoming DATA packets. DATA packet arrivals at different nodes are independent, and they arrive at the nodes with the same distribution. A node randomly selects one of its neighbors as the destination to transmit DATA packets for a certain time. To exclude influences from other than the investigated duty-cycled MAC protocol, routing is not considered.

4.2.2 Throughput Analysis

Throughput is defined as the amount of data successfully delivered within a unit time. Since the protocols work in a duty-cycled fashion, the throughput can be calculated within a cycle time. Therefore, the throughput of the system can be calculated as follows.

$$THR = N \cdot (1 - \pi_{emptyQ}) \cdot p_s \cdot S/T \tag{4.26}$$

Since the number of nodes in the network N, the MAC layer DATA packet size S, and the length of a cycle T are known, once the stationary probability of the empty-queue state π_{emptyQ} is solved by (4.8) and (4.24) or (4.23) and (4.25), the only unknown variable in (4.26) is the probability for each node to successfully transmit a DATA packet, p_s.

For no retransmission Markov model, p_s can be obtained by the similar way of obtaining p as a function of π_0 according to the media access rules of the investigated protocol. For retransmission Markov model, p_s can be obtained by equation (4.25). Hence, plugging the value π_{emptyQ} and p_s, the throughput of the network can be determined.

4.2.3 Delay Analysis

The delay of a DATA packet can be divided into two parts. The first part is the queuing delay D_Q, which is defined as the time interval from when a DATA packet joins the queue at the tail to the DATA packet becoming the head of the queue. The second part is the contending delay D_C, which is defined as the time interval from when the DATA packet is at the head of the queue to when the DATA packet is transmitted and hence removed from the queue. Therefore,

$$D = D_Q + D_C \tag{4.27}$$

The queuing delay of a DATA packet D_Q is the time that the DATA packet must wait in the queue until all the DATA packets in front of it finish contending for the media. Specifically, a newly joined DATA packet has to wait for a contending delay D_C for each of the DATA packets that are in front of it but behind the head of the queue. However, the newly joined DATA packet may arrive at the queue when the DATA packet at the head of the queue (if the queue is not empty) has already started contending for the media. Hence, for the DATA packet at head of the queue, the newly joined DATA packet has to wait on average for a half of the contending delay D_C of a DATA packet. According to the stationary probability π of our proposed Markov model, the queuing delay D_Q of a DATA packet can be calculated as a function of the contending delay D_C of a DATA packet as follows.

$$D_Q = D_C \cdot \sum_{i=0}^{Q-1} (max(0, i - 0.5) \cdot \pi_i / (1 - \pi_Q)) \tag{4.28}$$

The contending delay of a DATA packet D_C can be calculated according to the stationary probability π of our extended Markov model. A node with a DATA packet to send contends for the media once in a cycle, until the node finally wins the contention. For each contention, the node has a probability of p to win, and a probability of $1 - p$ to lose. Specifically, the node has a probability of p_s to win the contention and successfully transmit the DATA packet, and it has a probability of p_f to win the contention but encounters a collision. Therefore, given a cycle length of T, for no retransmission Markov model,

$$D_C = T \cdot \sum_{i=0}^{\infty} (i + 1) \cdot p \cdot (1 - p)^i \tag{4.29}$$

For retransmission Markov model,

$$D_C = T \cdot p_s \cdot \sum_{i=0}^{R} \sum_{j=0}^{\infty} \binom{i + j}{i} \cdot p_f^i \cdot (1 - p)^j \tag{4.30}$$

Plugging (4.29) or (4.30) into (4.28), then plugging (4.28) and (4.29) or (4.30) into (4.27), the queuing delay and the total delay of a DATA packet can be obtained.

4.2.4 Energy Consumption Analysis

Since we are considering duty-cycled MAC protocols, the average energy consumption per second of a node P can be obtained by calculating the energy consumption of a node in a cycle E divided by the cycle length T, i.e.,

$$P = E/T \tag{4.31}$$

However, the energy consumption of a node in a cycle, E, varies as the node plays different roles in the contention. (1) A node could be a sender that successfully transmits a data packet. According to the Markov model, the probability for a node to be a successful sender is $(1 - \pi_{emptyQ}) \cdot p_s$. (2) A node could be a receiver that successfully receives a data packet. Since every sender randomly selects a destination from its neighboring nodes, a node has the same probability of being a successful sender and being a successful receiver, i.e., $(1 - \pi_{emptyQ}) \cdot p_s$. (3) A node could be a sender that encounters a collision. According to the Markov model, the probability for each node to be an unsuccessful sender is $(1 - \pi_{emptyQ}) \cdot p_f$, where $p_f = p - p_s$. (4) A node could be a prospective receiver, but fails to receive the data packet due to a collision. As before, the probability for each node to be an unsuccessful receiver is the same as the probability of being an unsuccessful sender, i.e., $(1-\pi_{emptyQ})\cdot p_f$. (5) A node could be idle for its entire active period, as no other nodes attempt to transmit data when the node is awake. The probability of this case can be determined by the media access rules of the protocol. (6) A node could be idle, but it goes to sleep before its active period expires, as some other nodes access the media. The probability of this case can also be determined by the media access rules of the protocol.

To calculate the energy consumption of a node in a cycle, we (1) first calculate the energy consumption of a node in all the above cases. (2) Then, the energy consumption of each case is multiplied by the corresponding probability. (3) The summation of the products obtained in step (2) is the energy consumption of a node in a cycle. The average energy consumption per second of a node can then be calculated according to (4.31).

The average energy consumption per second of a node P can also be used to estimate the lifetime of the network. Assuming each node in the network has an initial energy E_{init}, the lifetime of the network can be represented as

$$L = E_{init}/P \tag{4.32}$$

4.3 Summary

In this chapter, we propose two Markov models to describe the queueing behavior of duty-cycled nodes with and without retransmissions. Meanwhile, we propose a general method to analyze the throughput, delay and energy consumption of the protocol using our proposed Markov models. Our Markov models and performance analysis method can be used for duty-cycled MAC protocols with fixed cycle length. In the next two chapters, we apply the proposed Markov models and performance analysis method to both synchronized and asynchronous duty-cycled MAC protocols. Our modeled results are validated via comprehensive simulations.

Chapter 5

Performance Analysis of a Synchronized MAC Protocol - S-MAC

In this chapter, we provide an example of how to apply our proposed Markov models and performance analysis methodology to S-MAC, a synchronized duty-cycled MAC protocol. We also show how to optimize protocol parameters using our proposed method to achieve desirable performance [28].

This chapter is organized as follows. Section 5.1 briefly introduces the media access rules of S-MAC. Section 5.2 analyzes the performance of S-MAC without retransmissions in fully-connected networks using our proposed no retransmission Markov model. Section 5.3 analyzes the performance of S-MAC with retransmissions in multihop networks using our proposed retransmission Markov model. Section 5.4 optimizes the performance of S-MAC using our proposed Markov models. Section 5.5 discusses the limitations of our Markov model, and performance analysis methodology, as well as our approach of model validation. Finally, section 5.6 summarizes the chapter.

5.1 S-MAC Overview

S-MAC [22] was the first duty-cycled MAC protocol designed for wireless sensor networks. It is also one of the most popular MAC protocols used for research on and implementation of wireless sensor networks [33][40]. S-MAC operates in a duty-cycled fashion, i.e., sensors sleep and wake up periodically, as shown in Fig. 5.1. The active period of a cycle has a fixed length, which is determined by the MAC layer contention window size. The sleeping period of a cycle, instead, could be shorter or longer, depending on the predefined duty cycle, which is the ratio of the active period length to the cycle length. All the nodes in the network have the same cycle length and duty cycle.

To improve the communication efficiency, S-MAC synchronizes sensors by exchanging their sleep-awake schedules in SYNC packets, so that every node sleeps and wakes up at the same time. A fixed interval in each active period is reserved for SYNC packet exchange. Specifically, a node broadcasts a SYNC packet periodically, for example, every 10 cycles. When a node is going to broadcast a SYNC packet, the node starts a random backoff during the SYNC period of the cycle. If the channel is free when the backoff procedure ends, the node broadcasts the SYNC packet with information about after how long the node is going to sleep in this cycle, assuming no data communication in this cycle. When other nodes receive this SYNC packet, they obtain the value of after how long the sender of the SYNC packet is going to sleep, and update their own sleep-awake schedules accordingly. Therefore, S-MAC synchronization can achieve an accuracy of 1 MAC layer clock cycle time resolution. For the CC2420 radio, IEEE 802.15.4 radio [25], the time resolution at the MAC layer is 0.1 ms.

S-MAC uses RTS/CTS/DATA/ACK handshaking to guarantee successful unicast transmissions. However, S-MAC has a fixed contention window, as a change in contention window size changes the length of an active period and hence influences the synchronization process. Moreover, when a node fails to win the contention or it encounters an RTS collision, it goes to sleep until the next active period. On the other hand, when a node sends out an RTS successfully, it does not go back to sleep until the transmitted DATA packet is acknowledged.

It is necessary to list all the reasons for DATA packet loss in S-MAC. Assuming ideal channels (i.e., no hidden terminals, capture effect or fading), a DATA packet could

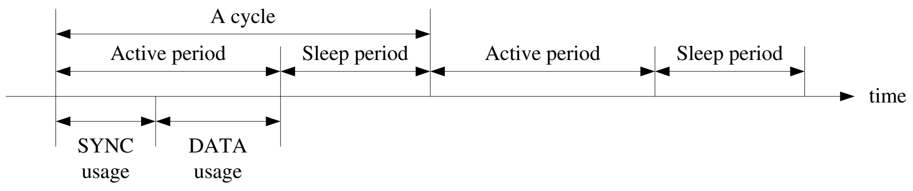


Figure 5.1: Sleep-awake cycle of S-MAC.

be dropped due to (1) overflow of the DATA packet queue, and (2) failure of the associated RTS (RTS collision) in the case of no retransmissions, and (3) over the limit of retransmission.

5.2 Performance Analysis of S-MAC Without Retransmissions

5.2.1 Throughput Analysis

Suppose there are N nodes in a fully connected network. When a node has a DATA packet to send, the probability that k out of the other $N-1$ nodes are competing for the media M_k can be described as a function of π_0.

$$\begin{aligned} M_k(\pi_0) &= \binom{N-1}{k} \cdot (1-\pi_0)^k \cdot \pi_0^{N-1-k} \\ &= \frac{(N-1)!}{k!(N-1-k)!} \cdot (1-\pi_0)^k \cdot \pi_0^{N-1-k}, \\ k &= 0..N-1 \end{aligned} \tag{5.1}$$

In the case that k other nodes are competing for the media, the probability of being the winner (sending out an RTS) p_k, and the probability of successfully sending out the DATA packet p_{sk} can be calculated as

$$p_k = \sum_{i=1}^{W} \frac{1}{W} (\frac{W-i+1}{W})^k, k = 0..N-1 \tag{5.2}$$

$$p_{sk} = \sum_{i=1}^{W} \frac{1}{W} (\frac{W-i}{W})^k, k = 0..N-1 \tag{5.3}$$

where W is the contention window size. Therefore,

$$p = g(\pi_0) = \sum_{k=0}^{N-1} M_k(\pi_0) \cdot p_k \tag{5.4}$$

$$p_s = h(\pi_0) = \sum_{k=0}^{N-1} M_k(\pi_0) \cdot p_{sk} \tag{5.5}$$

Solving (4.8) and (5.4), the probability of winning the contention p and the stationary probability of the empty-queue state π_0 can be obtained. Then, plugging the obtained

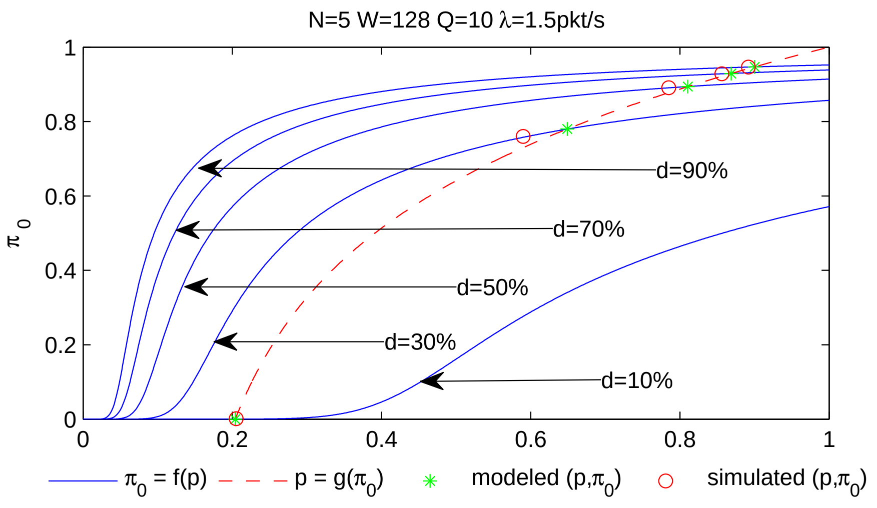


Figure 5.2: Determining (p, π_0) using $\pi_0 = f(p)$ and $p = g(\pi_0)$ for S-MAC.

π_0 into (5.5), the probability for each node to successfully transmit a DATA packet, p_s, can be determined. Finally, the network throughput of S-MAC can be calculated using (4.26). Fig. 5.2 shows an example of (4.8) and (5.4). Equation (4.8) is obtained from the Markov model, and is shown by the solid blue lines for different duty cycles. Equation (5.4) is obtained from the media access rules of S-MAC, and is shown by the dashed red line. The intersections of these curves, marked by the green asterisks, are the solutions of (4.8) and (5.4) for different duty cycles. Each solution corresponds to a specific (p, π_0), which is used to obtain the network throughput using (5.5) and (4.26).

5.2.2 Delay Analysis

According to (4.27)-(4.29), to calculate the delay of S-MAC, the probability for each node to win the contention, p, and the stationary distribution of the Markov model, π, need to be determined. p can be obtained by solving (4.8) and (5.4) together. Plugging the obtained p into (4.1)-(4.6), the stationary distribution of the Markov model π can be solved. Then, the delay of S-MAC can be obtained as described in section 4.2.3.

5.2.3 Energy Consumption Analysis

We define the power for transmitting data, receiving data and sleeping at each node as txp, rxp, and sp, respectively. Note that being idle consumes as much power as receiving data, since a node has to listen to the media. In this chapter, $txp = 52.2\ mW$, $rxp = 59.1\ mW$, $sp = 0\ mW$ according to the MICAz datasheet [73]. These power values are also used in the X-MAC energy analysis in Chapter 6.

To calculate the average energy consumption per second at each node, first it is necessary to recall some S-MAC operations. (1) Each node remains awake during T_{sync} in a cycle to send and receive SYNC packets, (2) each node transmits a SYNC packet every N_{sync} cycles, which is defined as a SYNC period, (3) each node remains awake for a SYNC period to avoid missing any SYNC packets from its neighbors every N_{awake} SYNC periods, hence, a node does not go to sleep in such awake cycles in contrast to going to sleep in the normal cycles, (4) T_{data} is defined as the longest time that a node may need to finish sending an RTS, however, RTS/CTS/DATA/ACK transmissions can last until a new cycle starts, and (5) in normal cycles, a node goes to sleep after transmitting/receiving a DATA packet successfully, or after experiencing an RTS/CTS collision, or after hearing an unintended RTS or CTS from its neighboring nodes.

As a result, a node may consume different amounts of energy in a normal cycle and in an awake cycle. Define E as the average energy consumption in a cycle at each node, E_{normal} as the average energy consumption in a normal cycle, and E_{awake} as the average energy consumption in an awake cycle. We have

$$\begin{aligned} E &= (E_{normal} \cdot N_{sync} \cdot (N_{awake} - 1) + E_{awake} \cdot N_{sync})/(N_{sync} \cdot N_{awake}) \\ &= (E_{normal} \cdot (N_{awake} - 1) + E_{awake})/(N_{sync} \cdot N_{awake}) \end{aligned} \tag{5.6}$$

Define E_{sync} as the energy consumption during T_{sync}, E_{data} as the energy consumption during data contention and data transmission, E_{sleep} as the energy consumption for sleeping during normal cycles, $E_{nosleep}$ and as the energy consumption for being awake after data contention and data transmission during awake cycles. Therefore,

$$E_{normal} = E_{sync} + E_{data} + E_{sleep} \tag{5.7}$$

$$E_{awake} = E_{sync} + E_{data} + E_{nosleep} \tag{5.8}$$

Assume each SYNC, RTS, CTS, DATA, and ACK packet takes t_{SYNC}, t_{RTS}, t_{CTS}, t_{DATA}, and t_{ACK} to transmit, respectively,

$$E_{sync} = ((t_{SYNC} \cdot txp + (T_{sync} - t_{SYNC}) \cdot rxp) + (T_{sync} \cdot rxp \cdot (N_{sync} - 1)))/N_{sync} \tag{5.9}$$

E_{data}, E_{sleep}, and $E_{nosleep}$ are related with the contention in the network, which determines how long a node needs to be active during random backoff before transmitting data or going to sleep. For a fully-connected network with N nodes, the number of contending nodes in a cycle N_0 has

$$E(N_0) = N \cdot (1 - \pi_0) \tag{5.10}$$

However, $E(N_0)$ may not be an integer, hence to simplify the calculation, we assume the two cases $N_0 = \lfloor E(N_0) \rfloor$ and $N_0 = \lceil E(N_0) \rceil$ have the highest probabilities and neglect the other possible values of N_0. Define $N_0^- = \lfloor E(N_0) \rfloor$ with a probability of p_-, and $N_0^+ = \lceil E(N_0) \rceil$ with a probability of p_+, we have

$$N_0^- \cdot p_- + N_0^+ \cdot p_+ = E(N_0) \tag{5.11}$$

$$p_- + p_+ = 1 \tag{5.12}$$

Plugging (5.10) into (5.11) and solving (5.11) and (5.12), p_- and p_- can be obtained. Consequently, we can define E_{data-} and E_{data+} as the energy consumption during data contention and data transmission when N_0^- and N_0^+ nodes are contending for the media, respectively.

$$E_{data} = p_- \cdot E_{data-} + p_+ \cdot E_{data+} \tag{5.13}$$

Similarly,

$$E_{sleep} = p_- \cdot E_{sleep-} + p_+ \cdot E_{sleep+} \tag{5.14}$$

$$E_{nosleep} = p_- \cdot E_{nosleep-} + p_+ \cdot E_{nosleep+} \tag{5.15}$$

When $N_0^- = 0$, it is easy to calculate E_{data-}, E_{sleep-}, and $E_{nosleep-}$, as no node is contending for the media in this case.

$$E_{data-} = T_{data} \cdot rxp \tag{5.16}$$

$$E_{sleep-} = (T - T_{sync} - T_{data}) \cdot sp \tag{5.17}$$

$$E_{nosleep-} = (T - T_{sync} - T_{data}) \cdot rxp \tag{5.18}$$

When $N_0^- > 0$ and hence $N_0^+ > 0$, a node has a probability $(1 - \pi_0) \cdot p_s$ of being the sender of a successful DATA transmission, implying it also has the same probability of being the destination of a successful DATA transmission. Similarly, a node has a probability $(1 - \pi_0) \cdot p_f$ of being the sender of an unsuccessful DATA transmission, implying it also has the same probability of being the destination of an unsuccessful DATA transmission. Assume (1) each packet has a propagation delay of D_{prop}, (2) W_{s-} and W_{s+} are the average backoff windows of the winner that successfully transmits a DATA packet in the case of N_0^- and N_0^+, respectively, (3) W_{c-} and W_{c+} are the average backoff windows of the winner that experiences a collision in the case of N_0^- and N_0^+, respectively, and (4) W_{t-} and W_{t+} are the average backoff windows of a winner of the contention in the case of N_0^- and N_0^+, respectively. Since the calculations for both N_0^- and N_0^+ are identical, to reduce the redundancy, only the calculations of E_{data-}, E_{sleep-}, and $E_{nosleep-}$ are shown as follows. To calculate E_{data+}, E_{sleep+}, and $E_{nosleep+}$, simply replace the subscript/superscript ”$-$” with ”$+$” in equations (5.19)-(5.24).

$$\begin{aligned}
&E_{data-} = \\
&(1-\pi_0)p_s((t_{RTS}+t_{DATA})txp + (t_{CTS}+t_{ACK})rxp + 4D_{prop}rxp + W_{s-}rxp) \\
&+(1-\pi_0)p_s((t_{RTS}+t_{DATA})rxp + (t_{CTS}+t_{ACK})txp + 3D_{prop}rxp + W_{s-}rxp) \\
&+(1-\pi_0)p_f(t_{RTS}txp + t_{CTS}rxp + 2D_{prop}rxp + W_{c-}rxp) \\
&+(1-\pi_0)p_f(t_{RTS}rxp + t_{CTS}txp + D_{prop}rxp + W_{c-}rxp) \\
&+(1-2(1-\pi_0)(p_s+p_f))(t_{RTS}rxp + W_{t-}rxp)
\end{aligned} \tag{5.19}$$

$$\begin{aligned}
&E_{sleep-} = \\
&(1-\pi_0)p_s(T - T_{sync} - W_{s-} - t_{RTS} - t_{CTS} - t_{DATA} - t_{ACK} - 4D_{prop})sp \\
&+(1-\pi_0)p_s(T - T_{sync} - W_{s-} - t_{RTS} - t_{CTS} - t_{DATA} - t_{ACK} - 3D_{prop})sp \\
&+(1-\pi_0)p_f(T - T_{sync} - W_{c-} - t_{RTS} - t_{CTS} - 2D_{prop})sp \\
&+(1-\pi_0)p_f(T - T_{sync} - W_{c-} - t_{RTS} - t_{CTS} - D_{prop})sp \\
&+(1-2(1-\pi_0)(p_s+p_f))(T - T_{sync} - t_{RTS} - T_{t-})sp
\end{aligned} \tag{5.20}$$

$$E_{nosleep-} =$$
$$(1-\pi_0)p_s(T - T_{sync} - W_{s-} - t_{RTS} - t_{CTS} - t_{DATA} - t_{ACK} - 4D_{prop})rxp$$
$$+(1-\pi_0)p_s(T - T_{sync} - W_{s-} - t_{RTS} - t_{CTS} - t_{DATA} - t_{ACK} - 3D_{prop})rxp$$
$$+(1-\pi_0)p_f(T - T_{sync} - W_{c-} - t_{RTS} - t_{CTS} - 2D_{prop})rxp$$
$$+(1-\pi_0)p_f(T - T_{sync} - W_{c-} - t_{RTS} - t_{CTS} - D_{prop})rxp$$
$$+(1-2(1-\pi_0)(p_s+p_f))(T - T_{sync} - t_{RTS} - T_{t-})rxp \tag{5.21}$$

However, to calculate (5.19), (5.20), and (5.21), the average backoff window of the contention winner that successfully transmits a DATA packet W_{s-}, the average backoff window of the contention winner that experiences a collision W_{c-}, and the average backoff window of the contention winner W_{t-} need to be obtained.

First, we calculate W_{s-}, the average backoff window of the contention winner that successfully transmits a DATA packet. For each possible backoff window from 0 to $W-1$, a node wins the contention and successfully transmits a DATA packet if all the other $N_0^- - 1$ nodes have larger backoff windows. Therefore,

$$W_{s-} = \sum_{i=0}^{W-1} i \cdot \left(\binom{N_0^-}{1} \cdot \frac{1}{W} \cdot \left(\frac{W-i-1}{W}\right)^{N_0^- -1} \Big/ \sum_{j=0}^{W-1} \left(\binom{N_0^-}{1} \cdot \frac{1}{W} \cdot \left(\frac{W-j-1}{W}\right)^{N_0^- -1}\right)\right) \tag{5.22}$$

Similarly, we calculate W_{c-}, the average backoff window of the contention winner that experiences a collision. For each possible backoff window from 0 to $W-1$, a node wins the contention but has a collision if (1) all the other $N_0^- - 1$ nodes have equal or larger backoff windows, and (2) at least one contending neighbor has the same backoff window. Therefore,

$$W_{c-} = \sum_{i=0}^{W-1} i \cdot \frac{\left(\frac{W-i}{W}\right)^{N_0^-} - \left(\frac{W-i-1}{W}\right)^{N_0^-} - \binom{N_0^-}{1} \cdot \frac{1}{W} \cdot \left(\frac{W-i-1}{W}\right)^{N_0^- -1}}{\sum_{j=0}^{W-1} \left(\left(\frac{W-i}{W}\right)^{N_0^-} - \left(\frac{W-i-1}{W}\right)^{N_0^-} - \binom{N_0^-}{1} \cdot \frac{1}{W} \cdot \left(\frac{W-i-1}{W}\right)^{N_0^- -1}\right)} \tag{5.23}$$

Finally, we calculate W_{t-}, the average backoff window of the contention winner. For each possible backoff window from 0 to $W-1$, a node wins the contention if all the other $N_0^- - 1$ nodes have equal or larger backoff windows. Therefore,

$$W_{t-} = \sum_{i=0}^{W-1} i \cdot \frac{\left(\frac{W-i}{W}\right)^{N_0^-} - \left(\frac{W-i-1}{W}\right)^{N_0^-}}{\sum_{j=0}^{W-1} \left(\left(\frac{W-j}{W}\right)^{N_0^-} - \left(\frac{W-j-1}{W}\right)^{N_0^-}\right)} \tag{5.24}$$

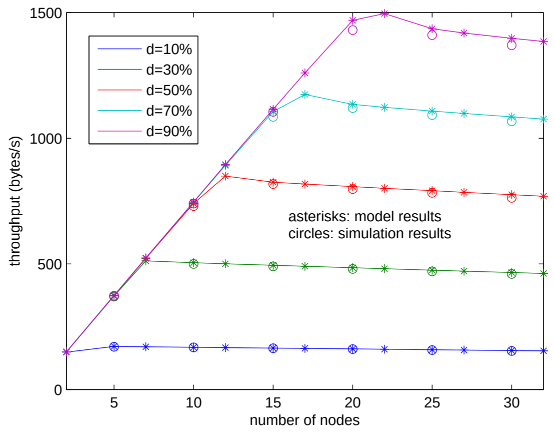
Plugging (5.22), (5.23) and (5.24) into (5.19), (5.20) and (5.21), E_{data-}, E_{sleep-}, and $E_{nosleep-}$ are obtained. Replacing subscript/superscript ”$-$” with ”$+$” in (5.19)-(5.24), E_{data+}, E_{sleep+}, and $E_{nosleep+}$ can be obtained. According to the value of N_0^- and N_0^+, substituting E_{data-}, E_{sleep-}, $E_{nosleep-}$, E_{data+}, E_{sleep+}, and $E_{nosleep+}$ into (5.16), (5.17), and (5.18), E_{data}, E_{sleep}, and $E_{nosleep}$ can be obtained. Calculating E_{sync} through (5.9), and substituting E_{sync}, E_{data}, E_{sleep}, and $E_{nosleep}$ into (5.7) and (5.8), and finally (5.6), the average energy consumption of a cycle at each node E is obtained. Therefore, the average energy consumption per second per node P can be obtained using (4.31).

5.2.4 Model Validations

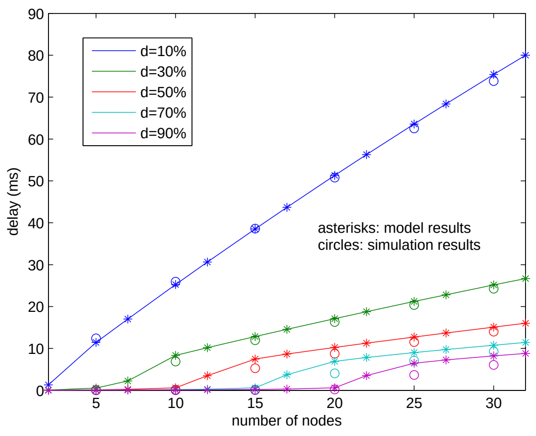
To validate our proposed no retransmission Markov model, we run the model under various S-MAC configurations and data arrival rates, and compare the estimates of throughput, delay, and energy consumption with simulation results using NS-2. In the simulation, (1) the network is fully-connected, (2) every 100 s one of a node's neighbors is randomly selected as the destination of the packets that arrive in the following 100 s, (3) the simulation time is 2000 s, and (4) all the results (throughput, delay, and energy consumption) are averaged over 50 runs.

In all the simulations, the basic set-up is a fully-connected network with $N = 15$ nodes, a contention window size W of 128, a data arrival rate λ at each node of 1.5 packets per second, and a queue capacity Q at each node of 10. For each set of simulations, we vary one of these parameters and investigate the throughput, delay and average energy consumption per second of S-MAC. Fig. 5.3 shows the performance results under varying node density from 5 nodes in the networks to 30 nodes in the network. Fig. 5.4 shows the performance results under varying data arrival rate λ from 0.5 packets per second to 2.5 packets per second at each node. Fig. 5.5 shows the performance results under varying contention window size W from 64 to 512. From these figures, we can see that our analytical results using the Markov model match the simulation results well for throughput and delay and approximate the simulation results for energy consumption.

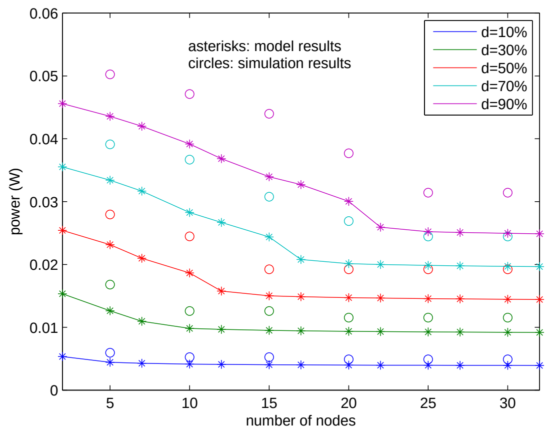
In Fig. 5.3a, the throughput of S-MAC first increases linearly as the number of nodes in the network increases, and then it stops increasing and decreases slightly as



(a) Throughput.

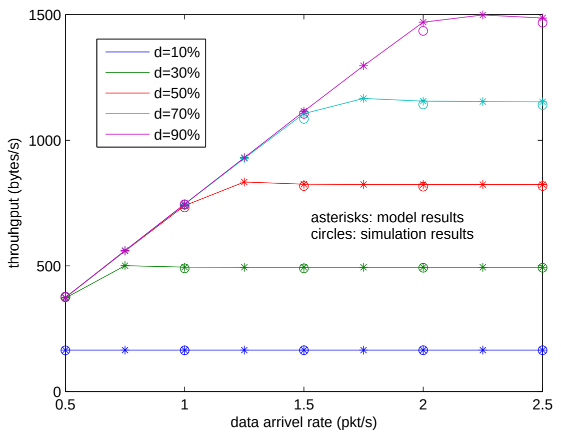


(b) Delay.

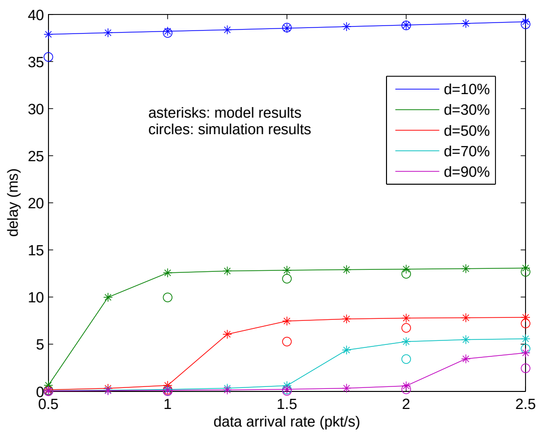


(c) Energy consumption/s/node.

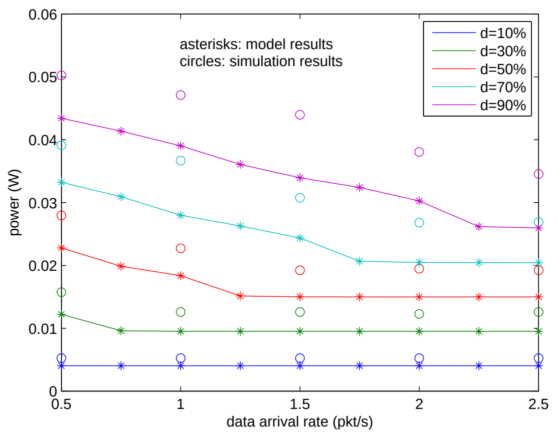
Figure 5.3: S-MAC performance with different numbers of nodes in the network.



(a) Throughput.

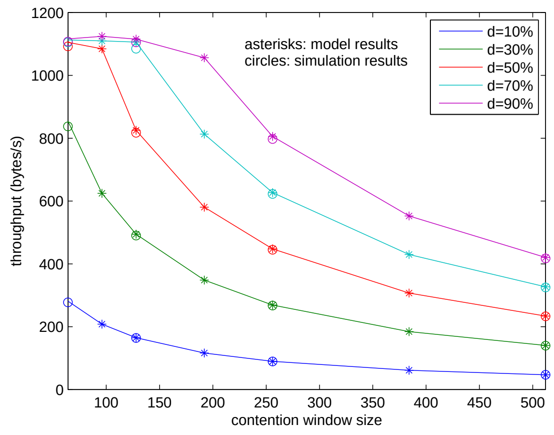


(b) Delay.

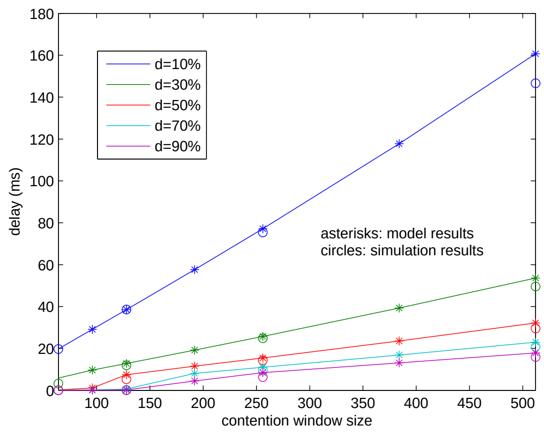


(c) Energy consumption/s/node.

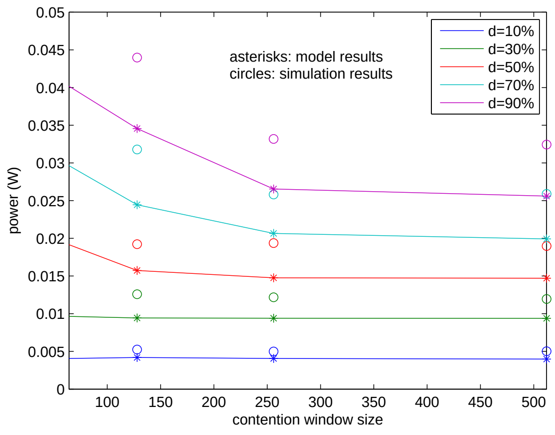
Figure 5.4: S-MAC performance with different data arrival rates.



(a) Throughput.



(b) Delay.



(c) Energy consumption/s/node.

Figure 5.5: S-MAC performance with different contention window sizes.

the node density further increases. Similarly, in Fig. 5.4a, the throughput of S-MAC first increases linearly as the data arrival rate increases, however, the throughput stops increasing and remains constant as the data arrival rate further increases. In Fig. 5.5a, the throughput of S-MAC first remains constant (S-MAC with 70% and 90% duty cycles), and then decreases as the contention window size increases. In all these figures, there is a change in the trend of throughput. Before the turning point of the throughput curves, S-MAC can deliver all the DATA packets as soon as they arrive in the network. Hence, the throughput increases linearly as the number of nodes increases, or the data arrival rates increases, and the throughput of S-MAC remains constant as the contention window size increases. However, after the turning point of the throughput curves, S-MAC saturates, i.e., S-MAC reaches its delivering limit and can no longer deliver all the incoming DATA packets, and thus DATA packets are backlogged in the queue until the queue overflows. As the node density keeps increasing, the contention in the network further increases. Hence, the probability of successfully transmitting a packet decreases, and more packets are finally dropped due to collisions and queue overflow. Consequently, the throughput decreases as the node density increases. Different from increasing the node density in the network, increasing the data arrival rate after S-MAC saturates does not increase the contention in the network since every node always has accumulated packets to send in each cycle. Therefore, more arriving packets are dropped due to queue overflow instead of collisions, and hence the throughput of S-MAC does not decrease. On the other hand, as the contention window size increases, the cycle length of S-MAC increases. Since S-MAC delivers at most one DATA packet per cycle, longer cycle length leads to lower throughput. Therefore, as the cycle length increases, the throughput of S-MAC decreases.

In Fig. 5.3b and Fig. 5.5b, the delay of S-MAC first remains nearly zero, and then increases as the node density increases or the contention window size increases. In Fig. 5.4b, the delay of S-MAC jumps from very low to a certain value as the data arrival rates increases. Similar to the throughput curves of S-MAC, the turning point in the delay curves of S-MAC is also due to the saturation of S-MAC. Before S-MAC saturates, S-MAC can deliver incoming packets as soon as they arrive at the network, hence few packets are accumulated in the queue, and as a result, the packet delay is very small. However, as S-MAC saturates, the amount of incoming data is more than what

S-MAC can deliver, hence the average queue length increases dramatically and then the queue overflows. As the node density increases (Fig. 5.3b), (1) the contention in the network increases, hence a packet has a longer contending delay before its associated RTS is sent, and (2) when S-MAC saturates, packets have a longer queuing delay as the average queue length increases. Therefore, the delay of S-MAC increases as the node density increases. For varying the data arrival rate (Fig. 5.4b), the increase in the packet delay is because the average queue length at each node increases dramatically as the data arrival rate increases. However, the average queue length cannot be longer than the queue capacity. Moreover, increasing the data arrival rate does not intensify the contention in the network after S-MAC saturates. Therefore, neither the contention delay nor the queuing delay of a packet increases. Hence, the packet delay remains constant after S-MAC saturates. For varying contention window size (Fig. 5.5b), after S-MAC saturates, all the nodes in the network have a packet to send in every cycle, and therefore the contention in the network (specifically the probability for each node to transmit a DATA packet p, and the stationary distribution of the Markov model π) remain almost constant as the contention window size increases. Since p and π are not sensitive to the contention window size after S-MAC saturates, according to (13) and (14) both contending delay and queuing delay increase linearly as the cycle length increases. Therefore, after S-MAC saturates the delay of S-MAC increases linearly as the contention window size increases.

In Fig. 5.3c, Fig. 5.4c, and Fig. 5.5c, the average energy consumption per node per second of S-MAC decreases as the number of nodes in the network increases, or the data arrival rate increases, or the contention window size increases. When there is little traffic in the network due to an extremely low node density or data arrival rate or contention window size, few nodes are contending for the media and all nodes are idle-listening to the channel during T_{data} in every normal cycle. As the node density or the data arrival rates increase, more nodes have packets to send in a cycle, and hence more nodes hear an unintended RTS and go to sleep before T_{data} expires. Meanwhile, as more nodes are contending for the media, the average backoff window of a contention winner decreases, which means nodes tend to hear an RTS earlier and go to sleep for a longer time in each cycle. Hence, on average, the energy consumption per second of each node decreases as the node density or the data arrival rate increases. When S-MAC saturates, a node can either be a contention winner or go to sleep after hearing an unintended RTS, but the chance of being awake and hearing nothing during T_{data} is small. For varying node density (Fig. 5.3c), as the node density increases, the average backoff window of a contention winner decreases slowly, which means less energy can be saved by hearing an earlier RTS. Therefore, the average energy consumption per second at each node decreases much slower than before. However, for varying data arrival rate (Fig. 5.4c), when S-MAC saturates, every node has a packet to send during each cycle, hence the average backoff window of a contention winner remains constant, and therefore, the average energy consumption per second cannot be decreased any more. For varying contention window size (Fig. 5.5c), a longer contention window means that unintended nodes have to to be awake for a longer time until the first RTS is sent. So, the energy consumption of a node per cycle increases as the cycle length increases. However, a longer contention window size also means a longer cycle length. In our experiment, the average energy consumption per second decreases slowly when S-MAC saturates.

Fig. 5.3, Fig. 5.4, and Fig. 5.5 also shows that for a given node density, S-MAC with a higher duty cycle has throughput and packet delay no worse than S-MAC with a lower duty cycle. However, S-MAC with a higher duty cycle always has higher energy consumption per second than S-MAC with a lower duty cycle, no matter whether S-MAC saturates or not.

5.3 Performance Analysis of S-MAC With Retransmissions in Multihop Networks

5.3.1 Throughput Analysis

When the nodes in the network are not in the communication range of each other, there are hidden terminals, which affect the relationship between $\pi_{(0,0)}$, p_s, and p_f. In this case, there could be more than one node in the network that wins the contention in each cycle. These winners send out RTSs, which stop their neighbors from contending the media. Hence, the calculation of p_s and p_f are not only determined by a node's 1-hop neighbors, but also determined by all the other winners throughout the network as a result of the network topology and the random backoff procedure at every node in the network. In general, it is hard to calculate p_s and p_f as a function of $\pi_{(0,0)}$ in the case of SMAC with hidden terminals.

However, it is obvious that the probability of successfully sending a DATA packet p_s and the probability of transmission failure of a DATA packet p_f are determined by the density of the contending nodes in the network, as well as the communication range of a node r. If the average number of contending nodes in the communication area of each node πr^2 is fixed, the contention in the network remains the same no matter the node density of the network or the communication range of each node changes. Hence, for a given contention window size W, p_s and p_f can obtained from simulations as a function of $d_c \cdot \pi r^2$. For a given node density d, $d_c = d \cdot (1 - \pi_{(0,0)})$. Therefore, once the stationary probability of the empty queue state $\pi_{(0,0)}$ is given, p_s and p_f can be determined accordingly.

The same as the calculations of p_s and p_f in the case of no hidden terminals, we assume that each node has an independent probability $1 - \pi_{(0,0)}$ to contend the media in the simulation of obtaining p_s and p_f as a function of $d_c \cdot \pi r^2$. Moreover, (1) the simulation simulates the contention in a given area of $6r \times 6r$ during a cycle time, (2) the number of contending nodes varies from 1 to 1000, hence the contending node density d_c varies from $1/(6r \cdot 6r)$ to $1000/()6r \cdot 6r$, (3) for a given d_c, 200 random topologies are generated, (4) for a given topology, each node that contends the media randomly selects a neighbor as its destination, and (5) to mitigate the edge effect, p_s and p_f are obtained by observing the nodes that are located in a circle with a radius r at the center of the given area. The simulation works according to the following algorithms.

```
Main algorithm:
Forcontending nodes = 1 to 1000
    Fortopology =1 to 200
        Algorithm 1: Simulation scenario generation
        Fortime slot = 0 to
            Fori=1 to contending nodes
                Algorithm 2: If node i's RTS timer expires
                Algorithm 3: If node i's CTS timer expires
            End
        End
```

```
      Algorithm 4: Calculate $p_s$ and $p_f$ for current simulation scenario
    End
    Calculate the average $p_s$ and $p_f$ for current $d_c \cdot \pi r^2$
End
```

```
Algorithm 1:
$d_c$ = contending nodes/(6r·6r)
Do
    Generate a new random topology with given number of contending nodes
    Each node randomly selects a neighboring node as its destination
Until every node has a neighboring node as its destination
Generate an RTS backoff time from 0 to $W - 1$ slots randomly for each node
Set CTS backoff time for each node to -1
Set RTS status for each node as on
Set CTS status for each node as N/A
Set Number of RTS sent as 0
Set Number of successful transmissions as 0
```

```
Algorithm 2:
If node i's RTS backoff timer expires and node i's RTS status is not off
   If node i is located in the observation circle in the center of the given area
      Increase the number of RTS sent by 1
   End If
   Checkevery neighboring node j of node i
           If node j's RTS expires within 2 time slots
              Set node i's RTS status as collision
              Set node j's RTS status as collision
           End If
           If node j's CTS expires within 2 time slots
              Set node i's RTS status as collision
              Set node j's CTS status as collision
           End If
```

```
            If node j's RTS expires after 2 time slots
               Set node j's RTS status as off
            End If
   End check
   If node i's RTS status if not collision
      Set node i's RTS status as sent
      Cancel the RTS backoff at node i's destination
      Set CTS backoff time at node i's destination as current time slot +2
   End If
End
```

Algorithm 3:

```
If node i's CTS backoff timer expires
   Checkevery neighboring node j of node i
            If node j's RTS expires within 2 time slots
               Set node i's CTS status as collision
               Set node j's RTS status as collision
            End If
            If node j's CTS expires within 2 time slots
               Set node i's CTS status as collision
               Set node j's CTS status as collision
            End If
            If node j's RTS expires after 2 time slots
               Set node j's RTS status as off
            End If
            If node j's CTS expires after 2 time slots
               Set node j's CTS status as off
            End If
   End check
   If node i's CTS status if not collision
      Set node i's CTS status as sent
      Increase the number of successful transmissions by 1
```

```
    End If
End
```

Algorithm 4:

Observed nodes = average number of contending nodes in the observation circle

p_s = number of RTS sent/Observed nodes

p_f = number of successful transmissions/Observed nodes

To avoid simulations for every S-MAC configuration, we further use the 5th inverse polynomial and the 5th logarithm to fit the simulation results of p_s and p_f as a function of $d_c \cdot \pi r^2$ for different contention window sizes, as shown in Table 5.1.

Table 5.1: Fitted Curves

W=64	
$p = a + b \cdot log(x) + c \cdot log^2(x) + d \cdot log^3(x) + e \cdot log^4(x) + f \cdot log^5(x)$	
$p_s = a_s + b_s \cdot log(x) + c_s \cdot log^2(x) + d_s \cdot log^3(x) + e_s \cdot log^4(x) + f_s \cdot log^5(x)$	
$a = 0.579430113353720$	$b = -0.226753470574262$
$c = 9.231459443895182e - 4$	$d = 0.007990973705467$
$e = 1.008095707617027e - 5$	$f = -1.54591671221983e - 4$
$a_s = 0.460090411010436$	$b_s = -0.253106905317230$
$c_s = 0.017237898851326$	$d_s = 0.010216617466037$
$e_s = -0.001432461496134$	$f_s = -8.64910081322444e - 6$

W=128	
$p = a + b \cdot log(x) + c \cdot log^2(x) + d \cdot log^3(x) + e \cdot log^4(x) + f \cdot log^5(x)$	
$p_s = a_s + b_s/x + c_s/x^2 + d_s/x^3 + e_s/x^4 + f_s/x^5$	
$a = 0.594833653961801$	$b = -0.251962553007999$
$c = -0.001714627073808$	$d = 0.016350359386391$
$e = -0.002566824260757$	$f = 8.633324494619900e - 5$
$a_s = -0.012892910195407$	$b_s = 0.893719239852677$
$c_s = -0.511288677638481$	$d_s = 0.154658883389399$
$e_s = -0.020620330721224$	$f_s = 9.140401561891618e - 4$

W=256	
$p = a + b/x + c/x^2 + d/x^3 + e/x^4 + f/x^5$	
$p_s = a_s + b_s/x + c_s/x^2 + d_s/x^3 + e_s/x^4 + f_s/x^5$	
$a = 0.012091383271150$	$b = 1.211400479758838$
$c = -0.862466394258291$	$d = 0.282689460344347$
$e = -0.038531919411938$	$f = 0.001716945384318$
$a_s = -0.008833286237725$	$b_s = 0.947493807187545$
$c_s = -0.595253367700030$	$d_s = 0.192610079718731$
$e_s = -0.026622420770892$	$f_s = 0.001201467813421$

W=512	
$p = a + b/x + c/x^2 + d/x^3 + e/x^4 + f/x^5$	
$p_s = a_s + b_s/x + c_s/x^2 + d_s/x^3 + e_s/x^4 + f/x^5$	
$a = 0.009693804144571$	$b = 1.202132578749658$
$c = -0.841409945036164$	$d = 0.278988820467379$
$e = -0.038541794955990$	$f = 0.001733415164259$
$a_s = -0.004963257312359$	$b_s = 0.959838267817747$
$c_s = -0.607602708058898$	$d_s = 0.197030443318555$
$e_s = -0.027152805806870$	$f_s = 0.001222431125310$

Using the fitted curves provided in Table 5.1, p and p_s can be represented as functions of $d_c \cdot \pi^2$. Since $d_c = d \cdot (1 - \pi_{(0,0)})$ and the node density d in the network is known, p and p_s can be determined for every given $\pi_{(0,0)}$. Meanwhile, p_f can be obtained by $p_f = p - p_s$. As a result, $\pi_{(0,0)}$ is represented as a function of p_s and p_f according to our extended Markov model, and p_s and p_f are functions of $\pi_{(0,0)}$ using the fitted curves. Solving these functions, we can obtain $\pi_{(0,0)}$, p_s and p_f. Plugging $\pi_{(0,0)}$ and p_s into the following equations, the throughput of S-MAC with retransmissions in multihop networks can be obtained.

$$THR = N \cdot (1 - \pi_{(0,0)}) \cdot p_s \cdot S/T \tag{5.25}$$

5.3.2 Delay Analysis

According to (4.27), (4.28) and (4.30), to calculate the delay of S-MAC, the probability for each node to successfully transmit a DATA packet p_s, the probability for each node to have a DATA transmission failure p_f, and the stationary distribution of the retransmission Markov model π, need to be determined. p_s and p_f can be obtained by solving (4.23) and (4.25) together. Plugging the obtained p_s a p_f into (4.9)-(4.21), the stationary distribution of the retransmission Markov model π can be solved. Then, the delay of S-MAC can be obtained as described in section 4.2.3.

5.3.3 Energy Consumption Analysis

The energy consumption per second per node for S-MAC with retransmissions in multihop networks can be obtained in the same way as obtaining energy consumption per second per node for S-MAC without retransmission in fully-connected networks, expect for a multihop network with N nodes, the average number of contending nodes in a cycle in a node's communication area N_0 is

$$E(N_0) = d_c \cdot (1 - \pi_0) \cdot \pi r^2 \tag{5.26}$$

This equation will take the place of (5.10). Following the analysis of S-MAC without retranmissions, the energy consumption per second per node for S-MAC with retransmissions in multihop networks can be determined.

5.3.4 Model Validations and Results

To validate the retransmission Markov model, we run the model under various S-MAC configurations and data arrival rates, and compared the estimations of throughput, delay, and power consumption the with simulation results using NS-2. In the simulation, (1) certain number of nodes are randomly distributed over a given area of 300 m $\times$ 300 m, (2) every node has a communication range of 50 m, (3) only the nodes that are located in the circle with 50m radius at the center of the given area are observed to mitigate the edge effect, (4) every node supports 3 retransmissions and has a queue capacity $Q = 10$, (5) every 100 s one of a node's neighbors is randomly selected as the destination of the packets that arrive in the following 100 s, (6) the simulation time is

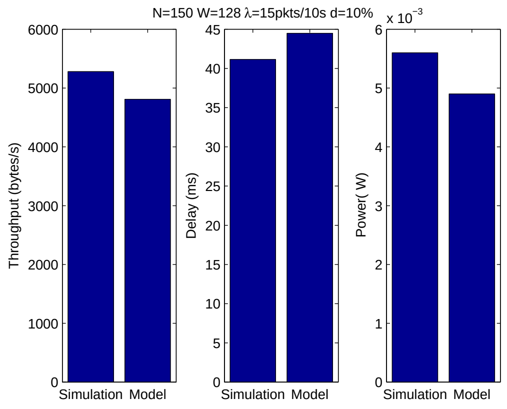
2000 s , and (7) all the results (throughput, delay, and energy consumption per second per node) are averaged over 50 randomly topologies.

Simulation results show that our retransmission Markov model well predicts the trend of throughput, delay and energy consumption per second per node of S-MAC. Fig. 5.6 gives a few examples of the comparisons between our modeled performance and simulated performance. In example 1, there are 150 nodes in a 300 m $\times$ 300 m area, the contention window size is 128, data arrival rate is 15 packets per 10 s, and the duty cycle is 10%. Our modeled throughput, delay and energy consumption per second per node approaches the simulated values with 8%, 8%, and 12% difference, respectively. In example 2, there are 150 nodes in a 300 m $\times$ 300 m area, the contention window size is 128, data arrival rate is 20 packets per 10 s, and the duty cycle is 50%. Our modeled throughput, delay and energy consumption per second per node approaches the simulated values with 1%, 12%, and 1% difference, respectively. In example 3, there are 150 nodes in a 300 m $\times$ 300 m area, the contention window size is 256, data arrival rate is 15 packets per 10 s, and the duty cycle is 30%. Our modeled throughput, delay and energy consumption per second per node approaches the simulated values with 8%, 14%, and 4% difference, respectively.

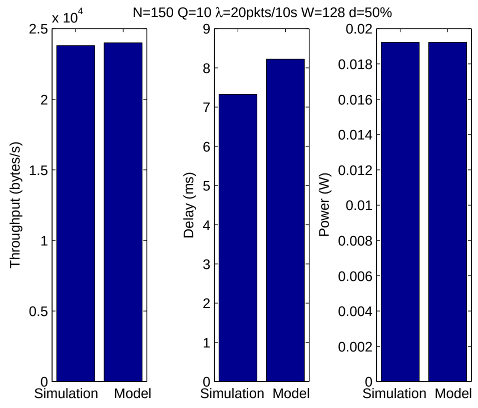
5.3.4.1 Varying the Number of Nodes

We first vary the node density in the network. Measured by the average number of nodes located in a node's communication range (a circle with a 50 m radius), the node density varies from 4.36 to 25.29, which can be converted to from 50 nodes to 290 nodes within a 300 m $\times$ 300 m area, as shown in Fig. 5.7. Meanwhile, the contention window size W is 128, the data arrival rate λ at each node is 15 packets per 10 s, the queue capacity Q at each node is 10, and the retransmission limit R is 3.

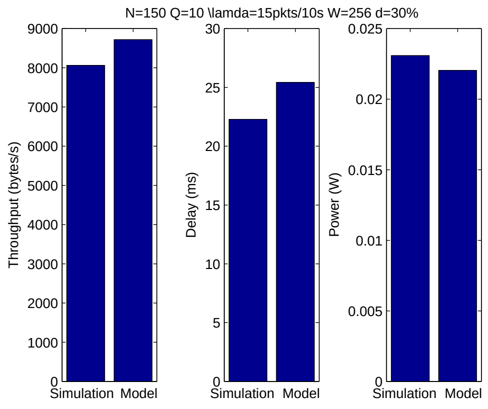
Fig. 5.7 shows the throughput, delay, and energy consumption per second per node of S-MAC under varying node density. First of all, there is a change in trend of the throughput, delay and power consumption. Specifically, as the node density increases, the throughput goes upward then downward, the delay stays steady and then increases linearly, and the power consumption decreases fast and then slows down. Second of all, for a given duty cycle, the curves of the throughput, delay, and the power consumption share the same turning point where the trend of the curve changes.



(a) Example 1.



(b) Example 2.



(c) Example 3.

Figure 5.6: Examples of model validations.

According to the above two observations, S-MAC saturates as the node density increases. Before the saturation point, which is the turning point of these curves for a given duty cycle, the contention in the network is so low that S-MAC can deliver a packet as soon as it arrives, hence the throughput increases linearly as the node density increases and there are few packets accumulated in the queue. However, after the saturation point, S-MAC reaches its delivery limit and packets start to accumulate in the queue at each node. As the node density keeps increasing, the contention in the network further increases. Hence, the probability of successfully transmitting a packet decreases, and more packets are finally dropped due to collisions and queue overflow. Consequently, the throughput decreases as the node density increases.

Similarly, the packet delay keeps steady before S-MAC saturates since S-MAC can deliver packets as soon as they arrive at each node. However, the packet delay increases linearly as S-MAC saturates. This is because (1) the contention in the network increases as the node density increases, hence, a packet has a longer contention delay before its associated RTS is sent, and (2) when S-MAC saturates, packets start to accumulate in the queue, hence packets have a longer queuing delay as the average queue length increases.

In terms of energy consumption per second per node, assume there is little traffic in the network due to an extremely low node density, no RTS is sent and nodes are idle-listening the channel during T_{data} in every normal cycle. As the node density increases, more RTSs are sent during T_{data}, and more nodes that are not the recipients but hear the RTSs go to sleep before T_{data} expires. Moreover, as the node density increases, the average backoff window of a contention winner decreases, which means nodes tend to hear an RTS earlier and go to sleep for a longer time. Hence, on average, the energy consumption per second of each node decreases as the node density increases. However, when S-MAC saturates, packets start to accumulate in the queue at each node. Initially, a node may have accumulated packets in its queue during some of the cycles but not every cycle. However, as the node density further increases, the probability at a node has accumulated packets in the queue becomes so high that every node has a packet to send during each cycle. As a result, the chance that a node does not hear any RTS (in a connected network) and hence has to be awake during T_{data} becomes so low that further increasing the node density does not help putting more nodes to sleep before T_{data} expires. Meanwhile, as the node density increases, the average backoff window of a contention winner decreases slowly, which means less and less energy can be saved by hearing an earlier RTS. Therefore, the energy consumption per second at each node decreases much slower than before. Note that the turning point of power consumption happens later than the turning points of throughput and packet delay, as the decrease of the energy consumption per second per node slows done when S-MAC saturates deeply.

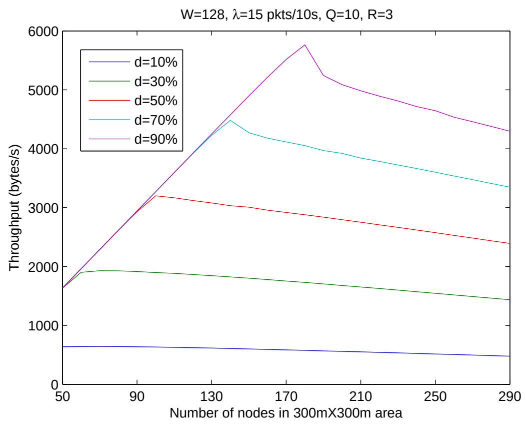
Fig. 5.7 also shows that for a given node density, S-MAC with a higher duty cycle has throughput and packet delay no worse then S-MAC with a lower duty cycle. However, S-MAC with a higher duty cycle always has higher power consumption than S-MAC with a lower duty cycle, no matter S-MAC saturates or not.

5.3.4.2 Varying the Data Arrival Rate

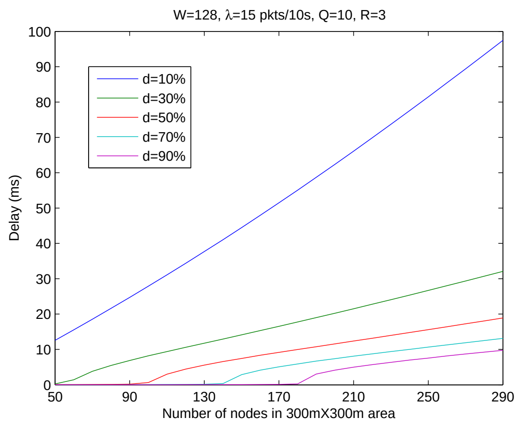
In this subsection, we vary the data arrival rate λ at each node in the network from 1 packet per 10s to 15 packets per 10 s. The node density is 13.08 nodes per communication range of a node, converting to 150 nodes within a 300 m $\times$ 300 m area. Meanwhile, the contention window size W is 128, the queue capacity Q at each node is 10, and the retransmission limit R is 3.

Fig. 5.8 shows the throughput, delay, and power consumption of S-MAC under varying data arrival rate. First, the throughput of S-MAC increases linearly as the data arrival rate increases. Until S-MAC can no longer deliver all the input packets as soon as they arrive in the network, S-MAC saturates and the throughput of S-MAC remains the same. Different with increasing the node density in the network, increasing the data arrival rate after S-MAC saturates does not intensify the contention in the network since every node always has accumulated packets to send in each cycle. Therefore, more arriving packets are dropped due to queue overflow instead of collision, and hence the throughput of S-MAC does not decrease.

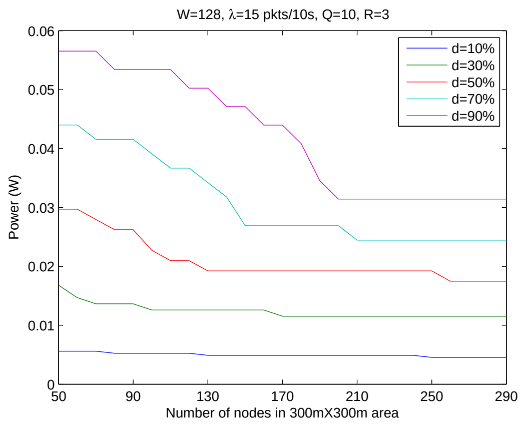
As the data arrival rate increases, the packet delay of S-MAC jumps from very low before S-MAC saturates to a certain value after S-MAC deeply saturates. The sharp increase in the packet delay is because the average queue length at each node increases dramatically as the data arrival rate increases. However, the average queue length cannot be longer than the queue capacity. Moreover, increasing the data arrival



(a) Throughput.



(b) Delay.



(c) Energy consumption/s/node.

Figure 5.7: S-MAC performance with different node density in multihop networks.

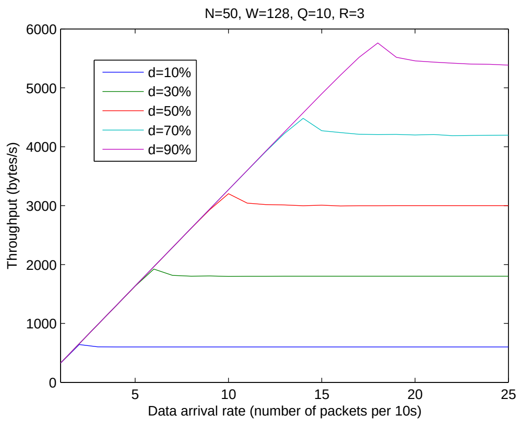
rate does not intensify the contention in the network after S-MAC deeply saturates. Therefore, neither the contention delay nor the queuing delay of a packet increases. Hence, the packet delay remains the same after S-MAC deeply saturates.

For a given duty cycle, the energy consumption per second per node of S-MAC starts to decrease from its maximum value at the lowest data arrival rate to its minimum value at a data arrival rate at which S-MAC deeply saturates. The maximum power consumption is reached when few nodes are contending the media and all nodes are idle-listening to the channel during T_{data} in every normal cycle. As the data arrival rates increases, more nodes have a packet to send in a cycle, and hence more nodes hear an unintended RTS and go to sleep before T_{data} expires. Meanwhile, as more nodes are contending the media, the average backoff window of a contention winner decreases, which means nodes tend to hear an RTS earlier and go to sleep for a longer time. Hence, on average, the energy consumption per second of each node decreases as the data arrival rate increases. The minimum energy consumption per second per node is reached when S-MAC deeply saturates, which means (1) every node has a packet to send during each cycle, hence the average backoff window of a contention winner reaches its lowest value, and (2) a node can either be a contention winner or go to sleep after hearing an unintended RTS, but the chance of being awake and hearing nothing during T_{data} is trivial. Therefore, the energy consumption per second per node of S-MAC cannot be decreased any more when S-MAC deeply saturates. This is different with Fig. 5.8 where the energy consumption per second per node slowly decreases due to a slowly decreasing average backoff window of a contention winner.

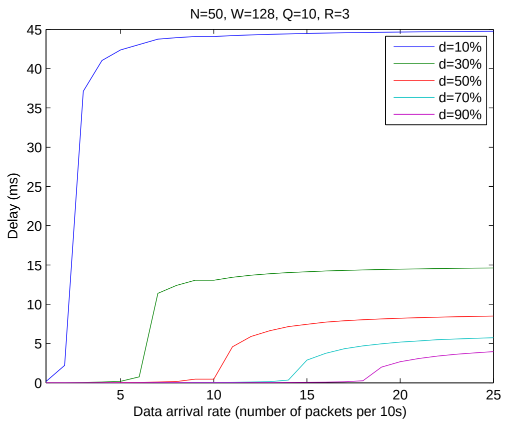
Similar with Fig. 5.7, Fig. 5.8 shows that for a given data arrival rate, S-MAC with a higher duty cycle has throughput and packet delay no worse then S-MAC with a lower duty cycle. However, S-MAC with a higher duty cycle always has higher energy consumption per second per node than SMAC with a lower duty cycle, no matter SMAC saturates or not.

5.3.4.3 Varying the Contention Window Size

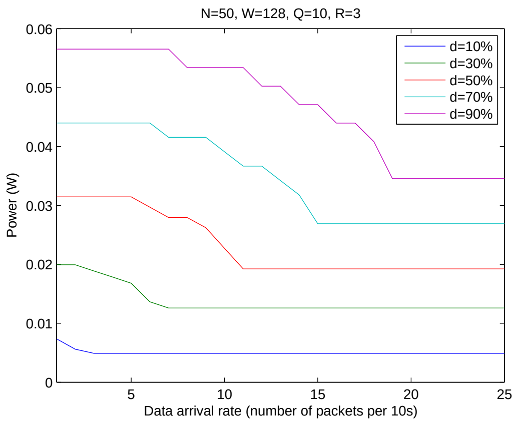
In this subsection, we vary the contention window size W at each node in the network from 64 time slots to 1024 time slots. The node density is 13.08 nodes per communication range of a node, converting to 150 nodes within a 300 m $\times$ 300 m area. Meanwhile,



(a) Throughput.



(b) Delay.



(c) Energy consumption/s/node.

Figure 5.8: S-MAC performance with different data arrival rates in multihop networks.

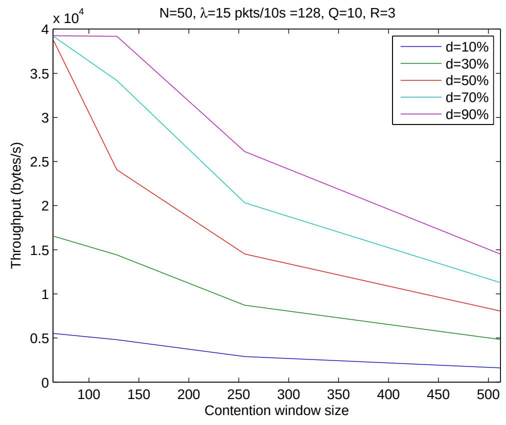
the data arrival rate λ is 15 packets per 10 s, the queue capacity Q at each node is 10, and the retransmission limit R is 3.

Fig. 5.9 shows the throughput, delay, and power consumption of S-MAC under varying contention window size. The throughput of S-MAC first remains the same, and then decreases as the contention window size increases. When S-MAC can deliver all the incoming DATA packets, the throughput of S-MAC is not affected by the variation of the contention window size. However, since the contention window size determines the cycle length of S-MAC, and S-MAC can deliver one DATA packet per cycle, the larger contention window size, the larger the cycle length, and hence the lower throughput when S-MAC saturates.

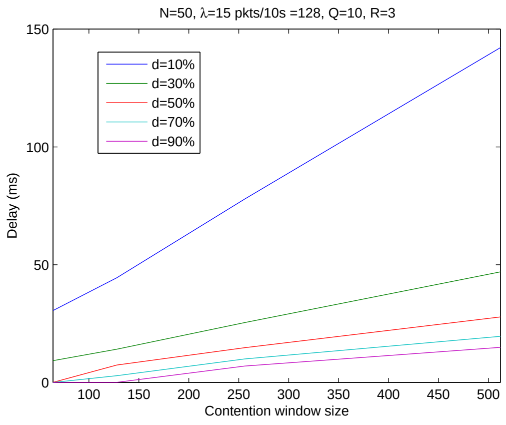
As the contention window size increases, the packet delay of S-MAC increases linearly. According to Chapter 4, the contenting delay and the queueing delay of S-MAC are in proportion to the cycle length. Therefore, as the contention window size increase, the cycle length increases, and the delay of S-MAC also increases.

For a given duty cycle, the energy consumption per second per node of S-MAC decreases as the contention window increases. For a particular number of contenting nodes in the network, the contention in the network does not change much due to the different contention window sizes. Hence, the energy consumed by the contention over a cycle time almost remains the same. However, as S-MAC delivers one DATA packet per cycle, the energy consumption for the DATA transmission over a cycle time decreases as the the cycle length increases along with the contention window size. Hence, the energy consumption per second per node of S-MAC decreases as the contention window size increases.

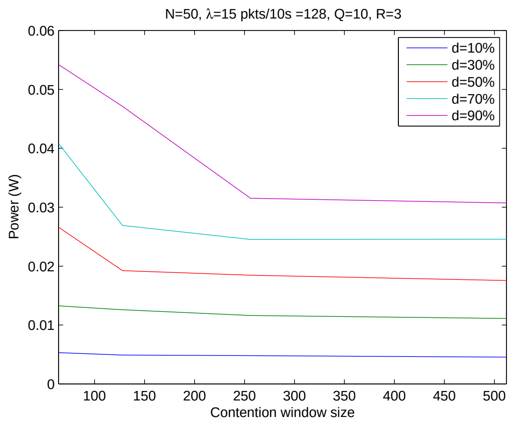
Similar with Fig. 5.7 and Fig. 5.8, Fig. 5.9 shows that for a given data arrival rate, S-MAC with a higher duty cycle has throughput and packet delay no worse then S-MAC with a lower duty cycle. However, S-MAC with a higher duty cycle always has higher energy consumption per second per node than SMAC with a lower duty cycle, no matter SMAC saturates or not.



(a) Throughput.



(b) Delay.



(c) Energy consumption/s/node.

Figure 5.9: S-MAC performance with different contention window sizes in multihop networks.

5.4 S-MAC Optimizations

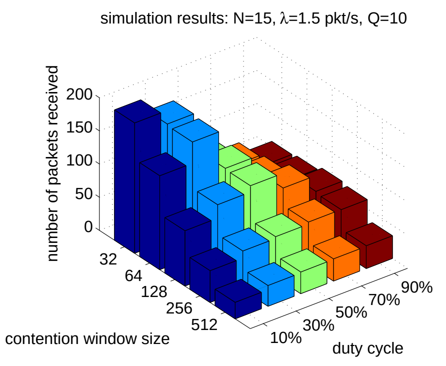
Our proposed Markov models can be used to estimate the throughput, delay, and energy consumption of S-MAC under different S-MAC configurations, network conditions and data arrival rates. Since S-MAC is the first, and the most basic, duty-cycled MAC protocol for wireless sensor networks, it is oftentimes used as a baseline in performance comparisons with newly proposed duty-cycled MAC protocols. Hence, future research on duty-cycled MAC protocols can benefit from the convenience and flexibility provided by our Markov model for S-MAC to avoid massive simulations.

Moreover, our proposed Markov models and performance analysis can be used to optimize the protocol parameters in order to achieve desirable performance, which may have requirements on throughput, delay, and energy consumption simultaneously. In this section, we show an example of optimizing the duty cycle and contention window size of S-MAC, in order to receive the most DATA packets in the lifetime of a network. The number of packets received at each node Pkt_{recvd} can be obtained by

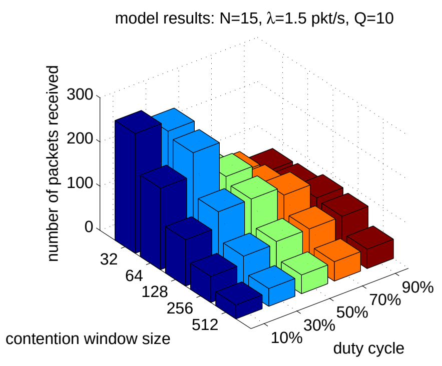
$$Pkt_{recvd} = THR/S/N \cdot L \tag{5.27}$$

where S is the DATA packet size, N is the number of nodes in the network, and L is the lifetime of each node in the network. Equation (5.27) can be obtained by (4.26) and (4.32) using our proposed Markov model and performance analysis.

Fig. 5.10 shows the number of packets a node can receive during its lifetime under different duty cycles and contention window sizes, using our no retransmission Markov model and simulations. Our model provides a good estimation of the optimal duty cycle and contention window size. In our experiment, there are 15 nodes in a fully-connected network, the data arrival rate is 1.5 packets per second at each node, and the queue capacity is 10. The available duty cycles are 10$\%$, 30$\%$, 50$\%$, 70$\%$, and 90$\%$, and the available contention window sizes are 32, 64, 128, 256, and 512. In order to receive the most DATA packets at each node, the optimal contention window is 32 and the optimal duty cycle is 10$\%$.



(a) Simulation Results.



(b) Analytical Results.

Figure 5.10: S-MAC performance optimization under different contention window sizes and duty cycles.

5.5 Discussion

In the model validation, the modeled performance obtained from our proposed Markov models follows the trend of the simulated performance under various network scenarios. However, the model performance approaches the simulated performance with certain offset. The offset is due to the limitations of our Markov models and performance analysis, as well as the simulation errors.

First of all, our Markov models assume that every node has the same and constant probability of winning the contention p in a cycle, or every node has the same and constant probability of successfully transmit a DATA packet p_s and probability of DATA transmission failure p_f. This assumption was used in previous work to analyze the performance of IEEE 802.11, IEEE 802.16 and IEEE 802.15.4 networks, and was proven to be a good approximation of the real case. However, this approximation incurs inaccuracy to the performance analysis, including our Markov models. To check the accuracy of this assumption in our no retransmission Markov model, we created a 2-D Markov model, assuming 2 nodes, each of which has different probability of winning the contention according to their queue sizes. Simulation results show that the probabilities for each node to win the contention are different, however, they are very close to each other. Therefore, our assumption that every node has the same and constant p is a good approximation of the real case. However, since p, p_s and p_f are key values in the calculations of throughput, delay and energy consumption, a small offset in p, p_s or p_f incurs non-negligible offset in the modeled performance to the simulated performance.

Secondly, some assumptions in our methodology to analyze the protocol's performance also cause inaccuracy in our modeled results. For example, we assume that every node contends for the media independently with a probability of $1 - \pi_{emptyQ}$. This assumption was also used in previous work on IEEE 802.11 networks [41]. However, nodes depend on each other to win or lose a contention, and hence they are not independent. For S-MAC with retransmissions in multihop networks, fitted curves are used to obtain the relationship between $\pi_{(0,0)}$ and p_s and p_f. The accuracy of the fitted curves can be further improved by increasing the number of simulations, increasing the number of different topologies in each simulation, enlarging the simulation area, and etc. Therefore, these assumptions contribute to the difference between our modeled results and the simulated results.

Meanwhile, in the energy consumption analysis of S-MAC, for a given average number of contending nodes in the cycle N_0 we only consider the two highly possible cases, $\lfloor E(N_0) \rfloor$ and $\lceil E(N_0) \rceil$, but neglect the other possible numbers of the contending nodes. This approximation simplifies the calculation, but incurs non-negligible errors in the modeled energy consumption per second per node.

Last but not the least, the simulated results are averaged over 50 runs of simulations, and hence they cannot be considered exact performance of a protocol. To make the simulated results more accurate, more simulations should be taken and more topologies in each simulation should be generated. However, in general, simulations cannot be used to obtain the exact performance of a protocol.

Due to the reasons above, our modeled performance approaches the simulated performance with certain offset. However, our Markov models can well predict the trend of the simulated performance, which is valuable in using our Markov model to optimize the protocol parameters.

5.6 Summary

In this chapter, we apply our general method to analyze the performance of duty-cycled MAC protocols to S-MAC, a synchronized MAC protocol for wireless sensor networks.

We validate our model using comprehensive simulations under various network conditions and data arrival rates. Specifically, we analyzed S-MAC without retransmissions in fully-connected networks and S-MAC with retransmissions in multihop networks. Results show that our method estimates well the the throughput, delay and energy consumption of S-MAC, and can be used to optimize the protocol parameters in order to achieve desirable performance. At the end of the chapter, we also discuss the limitations of our method and possible ways to improve the accuracy.

Chapter 6

Performance Analysis of an Asynchronous MAC Protocol - X-MAC

In this chapter, we apply our proposed no retransmission Markov model and performance analysis to X-MAC, an asynchronous duty-cycled MAC protocol for wireless sensor networks. We also show how to optimize protocol parameters using our proposed method to achieve desirable performance [29].

This chapter is organized as follows. Section 6.1 briefly introduces the media access rules of X-MAC. Section 6.2 analyzes the performance of X-MAC without retransmissions in fully-connected networks using our proposed no retransmission Markov model. Section 6.3 validates our Markov model using simulations. Section 6.4 optimizes the performance of X-MAC, and finally, section 6.5 summarizes the chapter.

6.1 X-MAC Overview

X-MAC avoids synchronization overhead, and hence it has higher energy-efficiency than synchronized MAC protocols such as S-MAC. Additionally, X-MAC uses a series of short preamble packets instead of an extended preamble, like B-MAC [23]. The short preamble packets carry the address information of the destination node. As a result, non-destination nodes can go to sleep as soon as they hear the first short preamble instead of remaining awake until the extended preamble ends. Moreover, the destination node can reply with an ACK in between two successive short preambles to stop the preamble and start the data transfer. Therefore, this strobbed preamble approach saves energy and greatly reduces latency. Furthermore, X-MAC has a fixed preamble size, and hence it can be readily adapted to the packetized radios that are emerging as the standard in today's sensor motes [74].

Fig. 6.1 shows the timeline of X-MAC. Every node in the network wakes up periodically to send and receive packets. The interval between two successive wake-ups is a cycle. X-MAC has a fixed cycle length for every node, yet each node starts its duty cycle with an arbitrary offset. As a result, when a sender wakes up to send a packet, the receiver may still be sleeping. Hence, the sender, from the time it wakes up, starts sending short preamble packets with the receiver's address information. In between two successive preamble packets, the sender pauses to listen to the media. At some point, the receiver wakes up and hears the preamble. The receiver sends an acknowledgement back to the sender during the pause between the two preambles. Note that the pause is shorter than the time that a node needs to detect an ongoing transmission, hence only the receiver can access the channel during the pause, while other nodes cannot interfere with the communication between the sender and the receiver [5]. When the sender receives the acknowledgement, it starts sending the DATA packet as the receiver is ready to receive. If a node wakes up without any packet to send, it goes to sleep if (1) the node hears an intended preamble packet, or (2) the node does not hear any transmission for a fixed amount of time, which is defined as the active time of a node, T_{active}, in a cycle.

Since X-MAC is asynchronous, for each DATA packet that is successfully delivered, the average communication time for the sender is $T/2$ plus the time to transmit a DATA packet t_{DATA} [74]. X-MAC also has collisions. When more than one sender wake up and start sending their preambles at the same time, all the other nodes, including the receivers, cannot determine the destination address information in the preambles. In this case, the senders will not stop sending preambles until their next wake-up time. Hence, for each colliding DATA packet, the average communication time for the sender is T [74]. For simplicity, in this chapter we analyze the throughput of slotted X-MAC. Hence, T, T_{active}, t_{DATA}, and other timing parameters in the following analysis are in the unit of a time slot, τ.

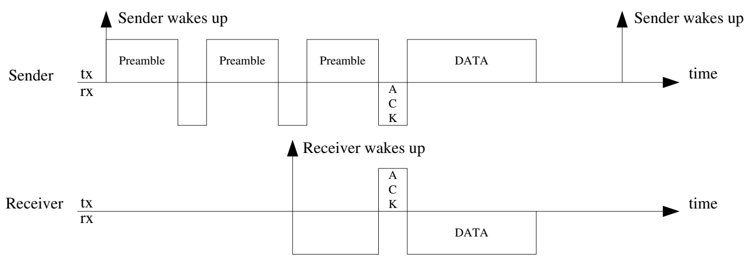


Figure 6.1: X-MAC timeline.

6.2 Performance Analysis of X-MAC

In this section, the no retransmission Markov model is used to analyze the throughput, delay and energy consumption of X-MAC in a fully-connected networks.

6.2.1 Throughput Analysis

Define p_f to be the probability of a collision when a node transmits a DATA packet in a cycle. Hence

$$p = p_s + p_f \tag{6.1}$$

We define (1) $empty$ and $\overline{empty}$ to be the status of the queue when a node wakes up. Hence the probability of having the queue "$empty$" is π_0 according to our proposed Markov model. (2) $free$ and $busy$ to be the status of the channel, and (3) A to be the event that a node becomes the only one winner of the contention, and B to be the event that a node becomes one of the multiple winners in the contention (implying a collision). Therefore, according to the Markov model, we have

$$p_s = Pr(A, free|\overline{empty}) = Pr(A|free, \overline{empty}) \cdot Pr(free|\overline{empty}) \tag{6.2}$$

$$p_f = Pr(B, free|\overline{empty}) = Pr(B|free, \overline{empty}) \cdot Pr(free|\overline{empty}) \tag{6.3}$$

We first solve for $Pr(A|free, \overline{empty})$ and $Pr(B|free, \overline{empty})$, and then we determine $Pr(free|\overline{empty})$.

Given a node has packets in its queue, and the channel is free when it wakes up, the node can successfully transmit a DATA packet as long as (1) no other nodes in the network wake up at the same time, or (2) some nodes wake up at the same time, but they have no packets to send. Hence,

$$Pr(A|free,\overline{empty}) = \sum_{t=1}^{T}\frac{1}{T}(\sum_{i=0}^{N-1}\binom{N-1}{i}(\frac{1}{T})^i{\pi_0}^i(\frac{T-1}{T})^{N-1-i}) \tag{6.4}$$

Similarly, given a node has packets to send in the queue, and the channel is free when it wakes up, the node has a collision when transmitting a DATA packet if at least one other node in the network wakes up at the same time and has packets to send. Hence,

$$Pr(B|free,\overline{empty}) = \sum_{t=1}^{T}\frac{1}{T}(\sum_{i=1}^{N-1}\binom{N-1}{i}(\frac{1}{T})^i(\sum_{j=1}^{i}(1-\pi_0)^j{\pi_0}^{i-j})(\frac{T-1}{T})^{N-1-i}) \tag{6.5}$$

$Pr(free|\overline{empty})$ is the probability of a free channel when a node wakes up with packets to send in its queue. Since in X-MAC every node wakes up and sends packets with an arbitrary offset to other nodes, the channel experiences the same probability to be $free$ or $busy$ in every time slot. Hence, when a single node wakes up, no matter whether its queue is empty or not, the node sees the channel with the similar probability of being $free$ or $busy$. Therefore,

$$Pr(free) \approx Pr(free|\overline{empty}) \tag{6.6}$$

This approximation is validated by comprehensive simulations using Matlab. Consequently, the problem of determining $Pr(free|\overline{empty})$ thus becomes the problem of determining $Pr(free)$.

The probability of a free channel, $Pr(free)$, can be obtained if the following two parameters are known: (1) the average length of a free channel between two transmissions over the media, E_{free}, and (2) the average length of a busy channel between the two chunks of a free channel, E_{busy}.

$$Pr(free) = \frac{E_{free}}{E_{free}+E_{busy}} \tag{6.7}$$

To calculate E_{free}, consider the time instant when a transmission ends and a chunk of free channel begins (note that the length of a specific chunk of free channel could be zero). From that time instant, the channel could be free for a certain number of cycles, say n cycles, until in the $n+1^{st}$ cycle, some node(s) start to transmit at the t^{th} slot.

The length of this chunk of free channel is $n \cdot T + t$, and the probability of this event $P_{free}(n,t)$ can be obtained as

$$P_{free}(n,t) = \pi_0^{Nn} \sum_{i=0}^{N-1} \sum_{j=1}^{N-i} \sum_{k=1}^{j}$$
$$\binom{N}{i} (\frac{t}{T})^i {\pi_0}^i \binom{N-i}{j} (\frac{1}{T})^j \binom{j}{k} (1-\pi_0)^k {\pi_0}^{j-k} (\frac{T-t-1}{T})^{N-i-j} \tag{6.8}$$

Therefore, the average length of a free channel between two transmissions over the media

$$E_{free} = \sum_{n=0}^{\infty} \sum_{t=0}^{T-1} (nT+t) \cdot P_{free}(n,t) \tag{6.9}$$

Similarly, if the channel is free for n cycles and t slots, the probability that a transmission is successful is $P_{busy}^{suc}(n,t)$, and the probability that a collision occurs is $P_{busy}^{col}(n,t)$.

$$P_{busy}^{suc}(n,t) = \pi_0^{Nn} \sum_{i=0}^{N-1} \sum_{j=1}^{N-i}$$
$$\binom{N}{i} (\frac{t}{T})^i {\pi_0}^i \binom{N-i}{j} (\frac{1}{T})^j \binom{j}{1} (1-\pi_0) {\pi_0}^{j-1} (\frac{T-t-1}{T})^{N-i-j} \tag{6.10}$$

$$P_{busy}^{col}(n,t) = \pi_0^{Nn} \sum_{i=0}^{N-2} \sum_{j=2}^{N-i} \sum_{k=2}^{j}$$
$$\binom{N}{i} (\frac{t}{T})^i {\pi_0}^i \binom{N-i}{j} (\frac{1}{T})^j \binom{j}{k} (1-\pi_0)^k {\pi_0}^{j-k} (\frac{T-t-1}{T})^{N-i-j} \tag{6.11}$$

The average length of a busy channel between the two chunks of a free channel E_{busy} can be calculated as the average length of a successful transmission $T/2 + t_{DATA}$ times the probability of a successful transmission, plus the length of a colliding transmission T times the probability of a collision. Hence,

$$E_{busy} = \sum_{n=0}^{\infty} \sum_{t=0}^{T-1} ((T/2 + t_{DATA}) \cdot P_{busy}^{suc}(n,t) + T \cdot P_{busy}^{col}(n,t)) \tag{6.12}$$

Plugging (6.9) and (6.12) into (6.7), we obtain $Pr(free)$. Then plugging (6.7) into (6.6), we obtain $Pr(free|\overline{empty})$. Plugging (6.4) and (6.6) into (6.2), and (6.5) and (6.6) into (6.3), the probability for each node to successfully transmit a DATA packet p_s, and the probability for each node to encounter a collision p_f are solved as a function of the stationary probability of the empty queue state π_0. According to (6.1), the probability for each node to transmit a DATA packet p can also be obtained as a function of π_0. Let function $g(\cdot)$ describe the relationship between p and π_0, we have

$$p = g(\pi_0) \tag{6.13}$$

Solving (4.8) and (6.13), the actual p and π_0 under which X-MAC is operating for a given scenario can be determined. Plugging in the actual π_0 to (6.2), the probability for each node to successfully deliver a DATA packet p_s can be obtained. The throughput of X-MAC can therefore be solved as follows according to (4.26) with a slight modification due to the time unit.

$$THR = N \cdot (1 - \pi_0) \cdot p_s \cdot S/(T \cdot \tau) \tag{6.14}$$

6.2.2 Delay Analysis

According to (4.27)-(4.29), to calculate the delay of X-MAC, the probability for each node to win the contention p, and the stationary distribution of the Markov model π need to be determined. p can be obtained by solving (4.8) and (6.13) together. Plugging the obtained p into (4.1)-(4.6), the stationary distribution of the Markov model π can be solved. Therefore, the delay of X-MAC can be obtained as described in Chapter 4, section 4.3 with a slight modification to (14).

$$D_C = T \cdot \tau \cdot \sum_{i=0}^{\infty}((i+1) \cdot p \cdot (1-p)^i) \tag{6.15}$$

6.2.3 Energy Consumption Analysis

Define an active period of a node in a cycle as T_{active} time units, and assume a preamble packet, a DATA packet, and an ACK packet take t_{pre}, t_{DATA}, and t_{ACK} time units to transmit, respectively.

For a fully-connected network with N nodes, a node has a probability $(1 - \pi_0) \cdot p_s$ of being the sender of a successful DATA transmission, which takes on average $T/2 + t_{DATA}$ to finish. During the $T/2 + t_{DATA}$ communication period, t_{DATA} is used to send the DATA packet, while the rest $T/2$ is used to periodically send preamble packets and listen to the ACK packets in between every two successive preamble packets. Hence, $T/2 \cdot (t_{pre}/(t_{pre} + t_{ACK}))$ is used to send preambles packets and $T/2 \cdot (t_{ACK}/(t_{pre} + t_{ACK}))$ is used to listen to the media. Therefore, the energy consumed in this case E_1 can be obtained as

$$\begin{aligned} E_1 = \ & (1 - \pi_0) \cdot p_s \cdot \tau \cdot \\ & (T/2 \cdot (t_{pre}/(t_{pre} + t_{ACK})) \cdot txp \\ & +T/2 \cdot (t_{ACK}/(t_{pre} + t_{ACK})) \cdot rxp \\ & +t_{DATA} \cdot rxp) \end{aligned} \tag{6.16}$$

Meanwhile, a node has a probability of $(1 - \pi_0) \cdot p_s$ being the destination of a successful DATA transmission. In this case, the node has to receive a complete preamble packet for t_{pre}, send an ACK packet for t_{ACK}, and receive the DATA packet for t_{DATA}. However, the receiving node may wake up when the transmitting node is half way through sending a preamble packet or listening for an ACK packet. As a result, the receiving node can only catch the next preamble packet sent by the transmitting node. Therefore, on average the receiving node listens to the media for $(t_{pre}+t_{ACK})/2$ before it receives a complete preamble packet. The energy consumed in this case E_2 can be obtained as

$$E_2 = (1-\pi_0)\cdot p_s\cdot\tau\cdot((t_{pre}+t_{ACK})/2\cdot rxp+t_{pre}\cdot rxp+t_{ACK}\cdot txp+t_{DATA}\cdot rxp) \tag{6.17}$$

Similarly, a node has a probability $(1-\pi_0)\cdot p_f$ of being the sender of an unsuccessful DATA transmission, implying at least one other node starts to send preambles at the same time. In this case, no preamble packets can be correctly received, and hence the senders keep transmitting preamble packets and listen to the media during the entire cycle T. Therefore, $T \cdot (t_{pre}/(t_{pre} + t_{ACK}))$ is used to send preamble packets and $T \cdot (t_{ACK}/(t_{pre} + t_{ACK}))$ is used to listen to the media in between two successive preamble packets. The energy consumed in this case E_3 can be obtained as

$$E_3 = (1-\pi_0)\cdot p_f\cdot\tau\cdot(T\cdot(t_{pre}/(t_{pre}+t_{ACK}))\cdot txp+T\cdot(t_{ACK}/(t_{pre}+t_{ACK}))\cdot rxp) \tag{6.18}$$

A node also has a probability $(1-\pi_0)\cdot p_f$ of being the destination of an unsuccessful DATA transmission. If the node wakes up when the colliding preamble packets are half way through transmission or when the colliding senders are listening to the media between two successive preamble packets, the node cannot detect the collision until it hears the next colliding preamble packets. Hence, on average the node keeps awake for $(t_{pre} + t_{ACK})/2 + t_{pre}$ before going to sleep. The energy consumed in this case E_4 can be obtained as

$$E_4 = (1 - \pi_0) \cdot p_f \cdot \tau \cdot ((t_{pre} + t_{ACK})/2 \cdot rxp + t_{pre} \cdot rxp) \tag{6.19}$$

At last, there is a probability of $1 - 2 \cdot (1 - \pi_0) \cdot (p_s + p_f)$ that a node is not involved in any data transmission in a cycle. The node goes to sleep if (1) it hears a complete preamble packet, or (2) T_{awake} expires. In this case, the energy consumption E_5 can be obtained as

$$\begin{aligned} E_5 = {} & (1 - 2 \cdot (1 - \pi_0) \cdot (p_s + p_f)) \cdot \tau \cdot rxp \cdot \\ & (\sum_{t=0}^{T_{active}-1} P_{free}(0, t) \cdot (t + (t_{pre} + t_{ACK})/2 + t_{pre}) \\ & +(\sum_{t=T_{active}}^{T-1} P_{free}(0, t) + \sum_{n=1}^{\infty} \sum_{t=0}^{T-1} P_{free}(n, t)) \cdot T_{active}) \end{aligned} \tag{6.20}$$

Overall, the energy consumption of a node in a cycle E can be obtained as the summation of E_1 to E_5.

$$E = \sum_{i=1}^{5} E_i \tag{6.21}$$

Hence, the average energy consumption per second of an X-MAC node P can be obtained according to (4.31).

6.3 Model Validations

To validate our proposed no retransmission Markov model, we run the model under various X-MAC configurations and data arrival rates, and compared the estimations of throughput, delay, and energy consumption per second per node with the simulation results using Matlab. In the simulation, (1) the network is fully-connected, (2) each node randomly selects one of its neighbors as the destination for every packet, (3) the simulation time is 1000 s, (4) all the results (throughput, delay, and energy consumption) are averaged over 50 simulations, and (5) the cycle length T is 200 ms. A time unit is

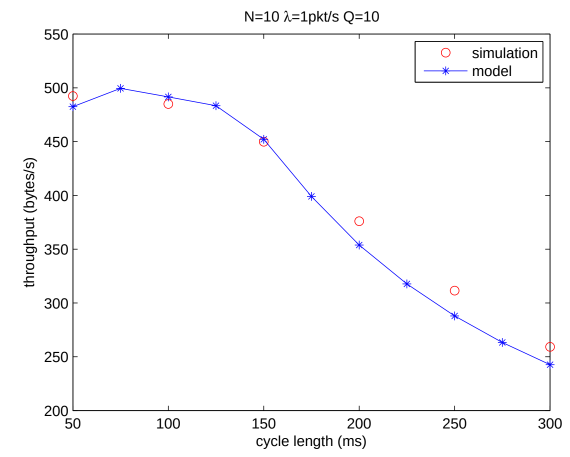
1 ms. The active time in each cycle T_{active} is 15. The time used to transmit a preamble packet t_{pre} is 3. The time used to transmit an ACK packet t_{ACK} is 1. The time used to transmit a DATA packet t_{DATA} is 5.

In all the simulations, the basic set-up is a fully-connected network with $N = 10$ nodes, a cycle length of $T = 200\ ms$, a data arrival rate λ at each node of 1 packet per second, and a queue capacity Q at each node of 10. For each set of simulations, we vary one of these parameters and investigate the throughput, delay and average energy consumption per second for X-MAC.

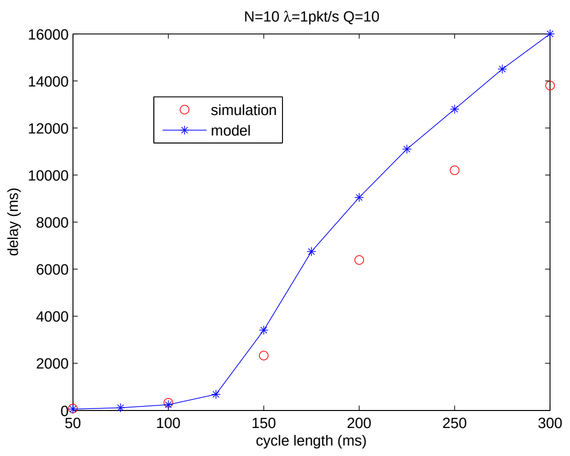
Fig. 6.2 shows the performance results under varying cycle length T from 50 ms to 300 ms. Fig. 6.3 shows the performance results varying the number of nodes N in the network from 5 to 40. Fig. 6.4 shows the performance results under varying data arrival rate λ from 0.5 packets per second to 5 packets per second. From these figures, we can see that our analytical results using the Markov model match the simulation results.

In Fig. 6.2a, as the cycle length increases, the throughput of X-MAC first remains the same as the amount of input data, and then decreases as X-MAC can no longer deliver all the incoming packets. Since X-MAC delivers at most one packet in a cycle, a longer cycle length leads to lower throughput. In Fig. 6.3a, before X-MAC saturates, the throughput of X-MAC increases as the number of nodes in the network increases. However, as the number of nodes increases further, the throughput of X-MAC saturates and then shrinks slightly due to the increasing collisions. In Fig. 6.4a, as the data arrival rate increases, the throughput of X-MAC first increases linearly as X-MAC can deliver more than the incoming traffic. However, as the data arrival rate increases further, the throughput of X-MAC saturates and remains the same. When X-MAC saturates, the queue at each node overflows. Hence, a node always has a packet to send whenever it wakes up. Since the number of nodes in the network is fixed, the contention in the network remains the same. As a result, the throughput of X-MAC does not change after saturation.

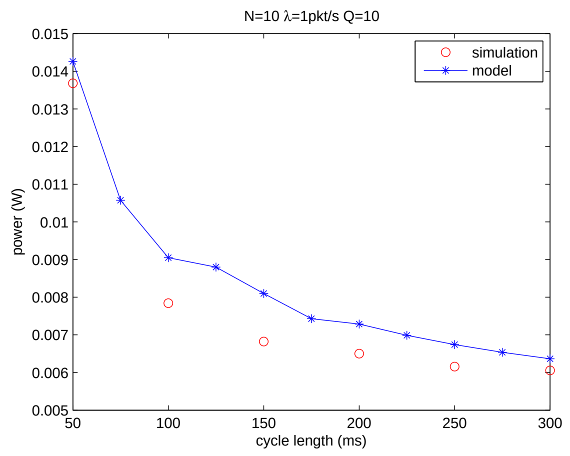
In Fig. 6.2b, when the cycle length is small, X-MAC can deliver all the incoming packets as soon as they arrive in the network, hence the delay of X-MAC is nearly zero. However, as the cycle length increases, X-MAC saturates and the queue at each node overflows. Both the contention delay and queuing delay, and hence the total delay of X-MAC, increases in proportion to the increase of the cycle length. In Fig. 6.3b, the



(a) Throughput.

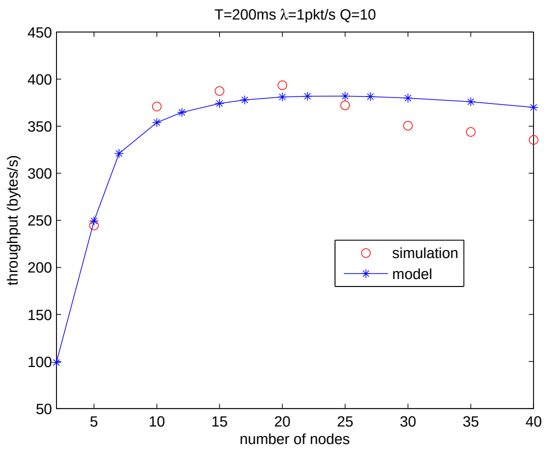


(b) Delay.

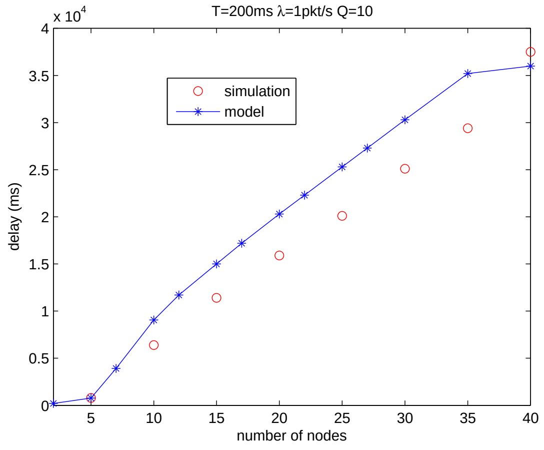


(c) Energy consumption/s/node.

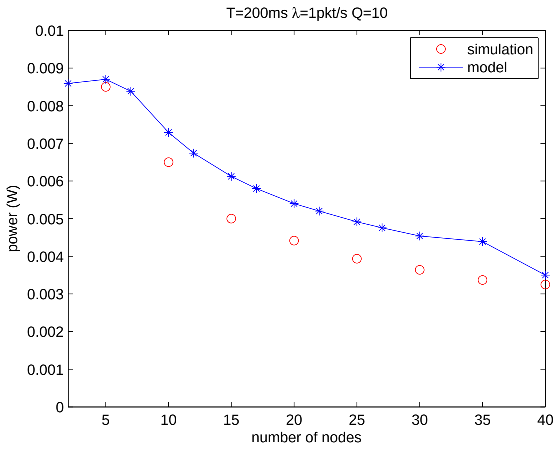
Figure 6.2: X-MAC performance with different cycle lengths.



(a) Throughput.

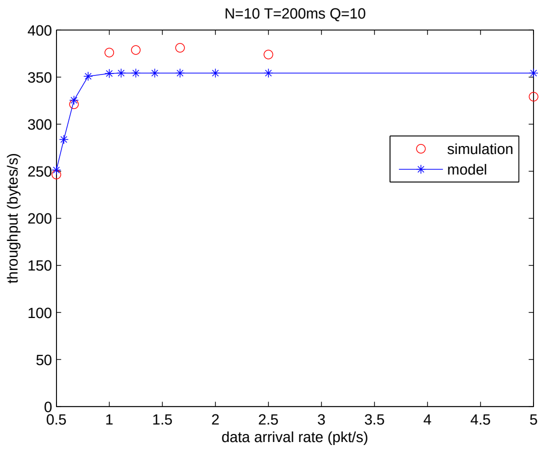


(b) Delay.

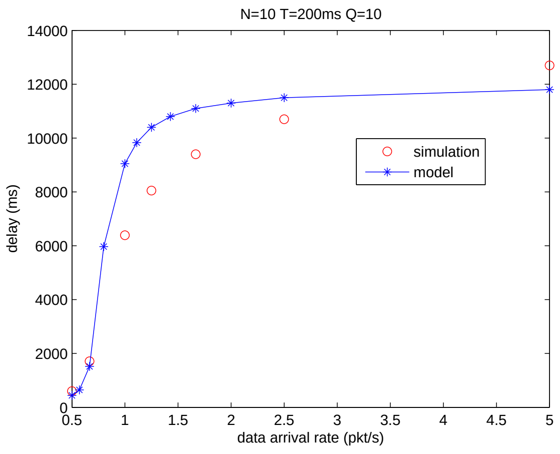


(c) Energy consumption/s/node.

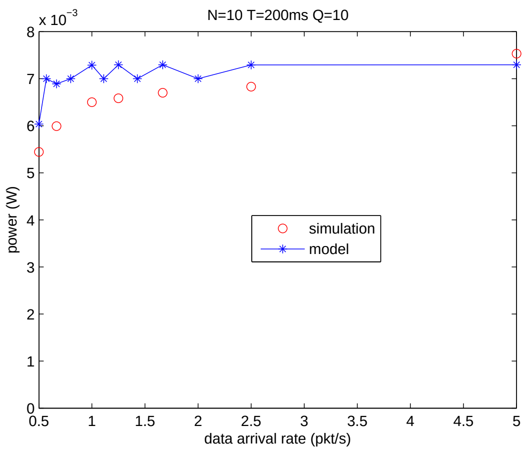
Figure 6.3: X-MAC performance with different numbers of nodes in the network.



(a) Throughput.



(b) Delay.



(c) Energy consumption/s/node.

Figure 6.4: X-MAC performance with different data arrival rates.

delay of X-MAC remains very small before X-MAC saturates, and then increases as the number of nodes increases. This is because the more nodes in the network, the lower the probability for each node to transmit a DATA packet in a cycle. Hence, according to (4.28) and (4.29), the contending delay and queuing delay of X-MAC increases. Moreover, as the number of nodes in the network increases, a node tends to have higher probabilities for higher queue lengths. As a result, the queuing delay of X-MAC further increases. Therefore, the delay of X-MAC increases as X-MAC saturates. In Fig. 6.4b, the delay of X-MAC increases as the data arrival rates increase. However, the increase is faster when X-MAC enters from the unsaturated region to the saturated region. This is because the queue at each node increases dramatically as X-MAC saturates. Hence, the increase in the delay of X-MAC is obvious. When the data arrival rates further increase, the increase of the delay slows down, as the queue length approaches the queue capacity.

In Fig. 6.2c, the energy consumption of X-MAC decreases as the cycle length increases. For a successful data transmission in X-MAC, the average communication time for the sender is $T/2 + t_{DATA}$. For an unsuccessful data transmission due to collision, the average communication time for the sender is T [74]. Hence, as the cycle length increases, the energy consumed in data transmission increases. However, since the active period in each cycle is fixed, all the nodes, except the sender and the receiver in the successful data transmission or the senders in the unsuccessful data transmission, go to sleep for a longer time as the cycle length increases. Since the energy savings in longer sleeping is more than the energy consumption in longer data transmissions, the energy consumption of X-MAC decreases as the cycle length increases. In Fig. 6.3c, X-MAC has decreasing energy consumption as the number of nodes in the network increases. X-MAC consumes energy mainly in data transmission (both successful and unsuccessful transmissions). Since X-MAC delivers at most one DATA packet in a cycle, more nodes in the network implies that a higher percentage of nodes fails to access the media and goes to sleep without consuming energy. Hence, the energy consumption of X-MAC decreases as the number of nodes increases. However, a larger number of nodes in the network also leads to a higher probability of collision, and hence more energy consumption. Therefore, as the number of nodes in the network increases, the decrease of the energy consumption slows down. In Fig. 6.4c, the average energy consumption per second first increases when X-MAC can deliver all the incoming packets. Therefore, as more packets are delivered, the average energy consumption per second increases. However, the average energy consumption per second remains the same after X-MAC saturates. As the data arrivals in the network exceed the X-MAC delivery capability, the queue at each node overflows. The higher the data arrival rate is, the more packets are dropped by the queue. Hence, neither the contention in the network nor the number of packets that are finally transmitted by X-MAC is sensitive to the data arrival rate. Therefore, the average energy consumption per second is limited when X-MAC saturates.

6.4 X-MAC Optimizations

Our proposed Markov model and performance analysis can be used to optimize some protocol parameters to achieve desirable performance. In this section, we show an example of optimizing the cycle length of X-MAC in order to receive the most DATA packets in the lifetime of a network. The number of packets received at each node can be obtained by (5.27).

Fig. 6.5 shows the number of packets a node can receive during its lifetime under different cycle lengths. Our model gives accurate prediction to the simulation results. For a fully-connected network with 10 nodes, if the data arrival rate is 1 packet per second at each node and the queue capacity is 10, the optimal cycle length is 150 in order to receive the most DATA packets at each node.

6.5 Summary

In this chapter, we analyze the throughput, delay, and energy consumption of X-MAC, an asynchronous duty-cycled MAC protocol, using our proposed no retransmission Markov model. Comprehensive simulation results validate our model under various network conditions and application requirements. Our Markov model together with the protocol-specific analysis can estimate well the throughput, delay and energy consumption per second per node for X-MAC, and it provides flexibility to achieve complex performance requirements by optimizing the protocol parameters.

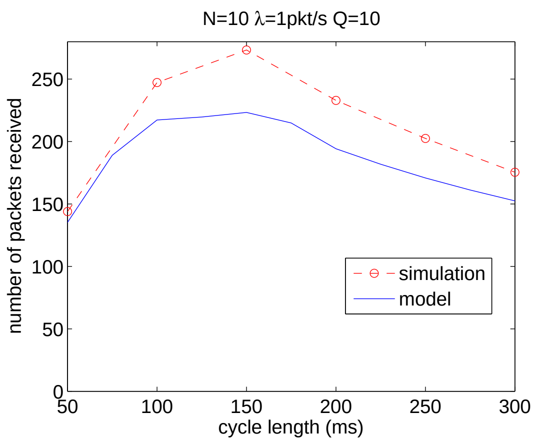


Figure 6.5: X-MAC performance optimization under different cycle lengths.

Chapter 7

Sleeping Multipath Routing: A Trade-off Between Reliability and Lifetime in Wireless Sensor Networks

Having studied and analyzed sleeping schemes at the MAC layer in the past few chapters, we now turn to sleeping at the routing layer, focusing specifically on analyzing the performance of sleeping multipath routing as a means of trading off reliability and network lifetime. Reliability is an issue in providing desirable QoS for applications that use wireless sensor networks. Since wireless channels are error-prone, and multihop communications are oftentimes necessary in various applications, data reliability over multihop transmissions may be significantly degraded. Multipath routing can greatly improve the reliability in wireless sensor networks. However, multipath routing also requires more nodes to be involved in the data delivery, implying more energy consumption and thus a shorter network lifetime.

To the best of our knowledge, sensor sleeping has not been introduced into multipath routing protocols to save energy and prolong the network lifetime. In this chapter, we propose Sleeping Multipath Routing, which selects the minimum number of paths to achieve a given reliability requirement and puts the rest of the network to sleep [30]. Sleeping Multipath Routing can be implemented to any multipath routing protocol that discovers multiple disjoint path between a sink node and a source node. Simulation results show that Sleeping Multipath Routing can significantly extend the network lifetime, and it can trade off reliability for lifetime.

The rest of this chapter is organized as follows. Section 7.1 elaborates on the idea of Sleeping Multipath Routing. Section 7.2 explains how to implement Sleeping Multipath Routing using Directed Diffusion. Section 7.3 details the benefits of Sleeping Multipath Directed Diffusion using simulations. Finally, section 7.4 summarizes the chapter.

7.1 Reliability vs. Lifetime

In this section, we first introduce an approach proposed by Dulman et al. [56] to provide reliability using multipath routing and FEC coding. Then, we propose our Sleeping Multipath Routing, a trade-off between reliability and lifetime, based on the approach in [56].

7.1.1 Reliability Support Using Multipath Routing and FEC Coding

The authors in [56] assume N disjoint paths from a source node to the sink node, discovered by a multipath routing protocol. A data packet is coded into N subpackets (including redundant subpackets), and each subpacket is transmitted over a different path. The sink node can reconstruct the data packet if at least K out of N subpackets are successfully delivered.

Each disjoint path i has a probability of successful data delivery p_i. Assuming the data delivery on each disjoint path is independent, the result of transmitting one packet on each disjoint path corresponds to a repeated Bernoulli experiment. Define K as the number of paths that successfully deliver the packet. We have

$$E(K) = \sum_{i=1}^{N} p_i \tag{7.1}$$

The distribution of the above repeated Bernoulli experiment can be approximated by a

Table 7.1: Value Pairs of α and x_α

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

normal distribution $N(\mu, \sigma^2)$ [56]. Specifically,

$$\mu = E(k) = \sum_{i=1}^{N} p_i \tag{7.2}$$

$$\sigma^2 = \sum_{i=1}^{N} p_i(1 - p_i) \tag{7.3}$$

Transforming the above normal distribution to a standard normal distribution, the random variable $K^* = \frac{K-\mu}{\sigma}$ is $N(0, 1)$-distributed. For a standard normal distribution and a given reliability requirement α, for example $\alpha = 95\%$ successful data reception, K^* has a lower bound x_α to guarantee the required reliability. Some value pairs of α and x_α are listed in Table 7.1. Therefore,

$$P(K^* \geq x_\alpha) \geq \alpha \Rightarrow P(K \geq \sigma x_\alpha + \mu) \geq \alpha \tag{7.4}$$

Since K is an integer and $K \geq 1$, we have

$$K = max\{\lfloor x_\alpha \sqrt{\sum_{i=1}^{N} p_i(1 - p_i)} + \sum_{i=1}^{N} p_i \rfloor, 1\} \tag{7.5}$$

which is the expected number of paths that successfully deliver the subpackets so that an overall reliability of α is achieved.

7.1.2 Sleeping Multipath Routing

To achieve a desired reliability, it may not be necessary to use all the disjoint paths, as illustrated in [56]. We therefore propose Sleeping Multipath Routing, which can prolong the network lifetime with reliability support by (1) carefully choosing multiple routing paths to achieve a given reliability, and (2) putting the rest of the sensors to sleep in order to save energy.

When $n < N$ disjoint paths can support the reliability requirement, all the nodes that are not on the n paths can go to sleep, saving energy for later use. When the activated n paths can no longer support the reliability (due to energy depletion, node failure, etc.), all the nodes in the network are activated, and the multipath routing protocol discovers new disjoint paths from the source node to the sink node. Similarly, if a subset of the new disjoint paths can support the reliability requirement, the rest of the network goes to sleep. Whenever the current activated paths expire, a new round of disjoint paths is discovered, until no set of paths that support the reliability requirement can be found.

Sleeping multipath routing can significantly extend the network lifetime. When no sleeping is allowed in the multipath routing protocol, all the nodes in the network have to be awake. As a result, every node has similar lifetime since wireless sensors consume similar power in transmitting data, receiving data, and being idle. Therefore, each disjoint path can deliver data for a similar amount of time, which is approximately the network lifetime.

However, Sleeping Multipath Routing can choose only a subset of the disjoint paths to achieve the application's reliability. Hence, if there exist multiple subsets of the disjoint paths that can support the application's reliability, these subsets of disjoint paths can serve the application one after another, and go to sleep when they are not activated by the Sleeping Multipath Routing protocol. As a result, the network lifetime can be prolonged multiple times depending on the available redundant subsets of disjoint paths. To maximize the network lifetime with reliability support, the minimum subset of disjoint paths (the subset that has the minimum number of disjoint paths) is chosen and activated when a new round of disjoint paths is discovered, so that more disjoint paths can sleep and serve the application in the future.

On the other hand, the minimum number of activated disjoint paths has to meet the reliability requirement. When N disjoint paths are discovered, reliability requirement α and corresponding x_α are known, the minimum number of disjoint paths that should be activated N_{min} can be obtained according to (7.5). We have

$$N_{min} = \begin{cases} 1 & \text{if } max(p_i) > \alpha, i \in [1..N] \\ min\ N^* & \text{otherwise} \end{cases} \tag{7.6}$$

$$\begin{aligned} s.t.\ \ & 2 \le N^* \le N \\ & K^* = \lfloor x_\alpha \sqrt{\sum_{i=1}^{N^*} p_{c_i}(1-p_{c_i})} + \sum_{i=1}^{N^*} p_{c_i} \rfloor \ge 1 \\ & c_i \in [1..N], i \in [1..N^*] \\ & \forall i \ne j, i \in [1..N^*], j \in [1..N^*], c_i \ne c_j \end{aligned}$$

Equation (7.6) implies that for each possible number N^* of activated disjoint paths, all the combinations of N^* different disjoint paths will be checked if there is a feasible K^* so that FEC coding can be used over the multiple paths to achieve the reliability requirement. The minimum N^* that can achieve the reliability requirement is N_{min}, the minimum number of disjoint paths that the sink node should finally activate.

However, it is computationally challenging to exhaustively check every combination of different N^* paths, determining whether it is feasible to apply FEC coding over these paths. To reduce the computation, Sleeping Multipath Routing sorts p_i, the probability of successful data delivery on disjoint path i, in descending order. For a given number N^* of disjoint paths, if the top N^* disjoint paths (N^* paths that have the highest probability of successful data delivery among the N paths) cannot make FEC coding feasible, it is impossible for other combinations of weaker paths to achieve the reliability requirement.

When the minimum number of activated disjoint paths N_{min} is determined, Sleeping Multipath Routing will activate the N_{min} disjoint paths that can (1) meet the reliability requirement, and (2) include the minimum number of sensors, to deliver data for the application.

In general, Sleeping Multipath Routing operates as follows. (1) Discover disjoint paths from the source node to the sink node. (2) Sort different disjoint paths in descending probability of successful data delivery. (3) Determine the minimum number of paths to activate according to (7.6). (4) Activate the minimum number of paths that can support the reliability requirement with the minimum number of sensors. (5) Put the rest of the sensors in the network to sleep. Sleeping nodes wake up periodically to receive routing updates. (6) When the activated paths can no longer support the reliability requirement, go to step (1).

Our Sleeping Multipath Routing is a general algorithm, which can be implemented to any multipath routing protocol that discovers multiple disjoint paths between the sink node and the source node.

7.2 Sleeping Multipath Routing: An Example Based on Directed Diffusion

In this section, we show an example of Sleeping Multipath Routing based on our proposed sleeping Directed Diffusion, described in Chapter 3. We first briefly revisit Directed Diffusion and sleeping Directed Diffusion, and then we elaborate on how to modify sleeping Directed Diffusion to support disjoint path discovery, multipath selection, and a reliability guarantee.

7.2.1 Directed Diffusion and Sleeping Directed Diffusion

Directed Diffusion includes two phases, namely exploratory phase and reinforcement phase, for a sink node to obtain data from a source node. In the exploratory phase, the sink node periodically broadcasts an INTEREST packet, which is then flooded throughout the network. When the source node receives the INTEREST packet, it starts broadcasting an exploratory DATA packet periodically. The exploratory DATA packet is flooded back to the sink node. Then the reinforcement phase starts. In the reinforcement phase, the sink node has a specific policy to reinforce some of the routing paths to pull down the data from the source node at high data rates. The sink node unicasts a POS_REINF packet to the selected neighbors. The nodes who receive the POS_REINF packet use the same policy to select their next hop to forward (unicast) the POS_REINF packet. When the source node receives the POS_REINF packet, it starts sending back DATA packets along the path where the POS_REINF packets came from. Therefore, the sink node finally obtains DATA packets at the high data rate from the reinforced paths.

Directed Diffusion does not support sensor sleeping. To save energy and prolong the network lifetime, we proposed a sleeping scheme for Directed Diffusion in [26], as shown in Fig. 7.1. Specifically, Directed Diffusion has two floodings, during which

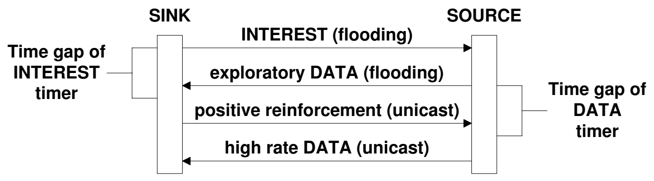

SINK
SOURCE
Time gap of INTEREST timer
INTEREST (flooding)
exploratory DATA (flooding)
positive reinforcement (unicast)
high rate DATA (unicast)
Time gap of DATA timer

Figure 7.1: Revisit of Sleeping Directed Diffusion [26].

all the nodes in the network have to be awake. One flooding is that the sink node floods an INTEREST packet periodically to initiate exploratory DATA flooding. The other flooding is that the source node floods an exploratory DATA packet periodically to start the reinforcement phase. When the unicast data delivery path(s) is established, the reinforced nodes keep awake while all the other nodes can go to sleep until the next INTEREST flooding or exploratory DATA flooding occurs.

Therefore, sleeping Directed Diffusion sets two timers at each node, namely an INTEREST timer and a DATA timer, corresponding to the two floodings in the network. Each timer is scheduled some time after the corresponding flooding is over, and it expires before the next corresponding flooding starts. Therefore, when the two timers are pending, no flooding is going on in the network. If the node does not receive a POS_REINF packet, it can go to sleep until any one of the two timers expires.

7.2.2 Sleeping Multipath Routing: Disjoint Path Discovery

Depending on the reinforcement policy, Directed Diffusion and sleeping Directed Diffusion can discover multiple braided paths from the source node to the sink node. However, disjoint paths are not guaranteed. We therefore modify sleeping Directed Diffusion in the following ways so that the reinforcement phase discovers only disjoint paths.

(1) Only the sink node can reinforce more than one neighbor, from which it receives exploratory DATA packets.

(2) All the nodes other than the sink node can only reinforce one neighbor at a time.

(3) If a node receives a POS_REINF packet but has no neighbor to reinforce (all neighbors have already been reinforced or no neighbor delivers an exploratory DATA to this node before), the node unicasts a CANCEL_POS packet to the sender of the POS_REINF packet.

(4) When a node receives a CANCEL_POS packet, it cancels the positive reinforcement to the sender of the CANCEL_POS packet, and selects another neighbor to reinforce. If the node finally finds no neighbor to reinforce, it unicasts a CANCEL_POS packet back to the node who sent the POS_REINF packet to it.

CANCEL_POS packets and the "reinforce one neighbor at time" policy guarantee that all the reinforced paths are disjoint. Therefore, when the source node receives the POS_REINF packets, DATA packets will be sent back to the sink node along the reverse paths.

7.2.3 Sleeping Multipath Routing: Multipath Selection

Sleeping Multipath Routing first discovers all the disjoint paths from the source node to the sink node. When the sink node receives high rate DATA packets from each disjoint path after sending out the POS_REINF packets, it has to select some of the disjoint paths to support the reliability requirement according to (7.6).

Every reinforced node estimates the probability of successful data reception over the link on which the data arrive. This estimation can be based on the data reception history or measured link quality. For a given disjoint path i, the probability of successful data delivery p_i is the product of the probability of successful data reception on every link of the disjoint path.

$$p_i = \prod_{e \in path\ i} prob_e \tag{7.7}$$

where e is a link in path i, and $prob_e$ is the probability of successful data reception over link e. When the source node generates a DATA packet, it sets the probability of successful data delivery to 1, and puts this information into the packet header. A node who receives a DATA packet updates the packet header by multiplying its own probability of successful data reception to the probability of successful data delivery, originally carried by the packet header. When the sink node receives a DATA packet, it obtains the probability of successful data delivery over the disjoint path along which the DATA packet was transmitted. Obtaining the probability of successful data delivery on each reinforced path, the sink node can decide which disjoint paths to finally use to serve the application, as described in section 7.1.2.

After multipath selection, the sink node has to "turn off" the paths that are reinforced but are not selected to serve the application. The sink node unicasts NEG_REINF packets along those reinforced but not selected paths. When the source node receives the NEG_REINF packets, it stops unicasting DATA packets along those paths. All the nodes that are negatively reinforced can go to sleep according to the two timers, which are defined in sleeping Directed Diffusion.

7.2.4 Sleeping Multipath Routing: Reliability Guarantee

Sleeping Multipath Routing has to avoid QoS pauses, so that the reliability is not affected by the handover between different disjoint paths. To avoid QoS pauses, the sink node estimates how much time each discovered disjoint path can sustain. Every node estimates how long it can be active according to its remaining energy and power. When the source node generates a DATA packet, it puts its remaining lifetime into the packet header. A node receiving a DATA packet will compare its own estimated remaining lifetime with the lifetime piggy-backed by the DATA packet. If the estimated remaining lifetime of this node is shorter than the piggy-backed lifetime, the node updates the DATA packet header with its own estimated remaining lifetime. Therefore, when the sink node receives DATA packets from multiple disjoint paths, it can estimate how long the path can be active. When the sink node realizes that one of the activated paths cannot be active more than one exploratory DATA period, it negatively reinforces all the reinforced paths after the next exploratory DATA flooding, and starts a new disjoint multipath discovery. In this way, a dying disjoint path will be negatively reinforced before it depletes its energy, and hence QoS pauses are avoided to ensure that reliability is met.

Another reliability issue is that the sink node has to inform the source node of the specific FEC code to use. This can be achieved by piggy-backing the FEC information through the NEG_REINF packets.

7.3 Simulation Results

We implemented our proposed Sleeping Multipath Routing based on Directed Diffusion in NS-2. In this section, we compare the Sleeping Multipath Directed Diffusion with no sleeping Multipath Directed Diffusion. Simulation results show that our proposed Sleeping Multipath Routing can trade off between reliability and lifetime in wireless sensor networks. In the simulation, (1) nodes are randomly placed in a 100 m x 100 m area, (2) there is one sink node and one source node in the network, (3) nodes have a transmit power of 52 mW, a receive power of 59 mW, an idle power of 59 mW, and a sleeping power of 0 mW [87], (4) the sink node and the source node have plenty of energy so that the network lifetime is not limited by the sink node or the source node, while other nodes have initial energy of 25 J, (5) the probability of successful data reception from one node to another is randomly generated between 85% to 100%, (6) the INTEREST period is 150 s, and the exploratory DATA period is 100 s, (7) the IEEE 802.11 MAC protocol is used, and (8) simulation results are averaged over 50 runs, each of which has a different random topology.

7.3.1 Varying the Reliability Requirement

In this experiment, we place 50 nodes in the 100 m x 100 m area. The reliability requirement varies from 80% to 95%. Fig. 7.2 shows the average lifetime, maximum lifetime and minimum lifetime of Sleeping Multipath Directed Diffusion (SMDD), compared to the average lifetime of no sleeping Multipath Directed Diffusion (MDD). As we can see, the average lifetime of MDD remains the same as the reliability requirement varies. However, the lifetime of SMDD is always longer than the lifetime of MDD, and it decreases as the reliability requirement increases. Tuning the reliability requirement enables our proposed Sleeping Multipath Routing protocol to trade off between reliability and lifetime by controlling the number of activated disjoint paths. Fig. 7.2 also shows that the minimum lifetime of SMDD is always no shorter than the lifetime of MDD, as in the worst case, SMDD cannot find new disjoint paths to support the reliability after the first disjoint paths die, and the lifetime of SMDD is the same as MDD. However, the maximum lifetime of SMDD can be many times longer than the lifetime of MDD. In general, a lower reliability requirement means that more disjoint paths can be discovered to support the reliability, and hence more routing redundancy is available to extend the network lifetime.

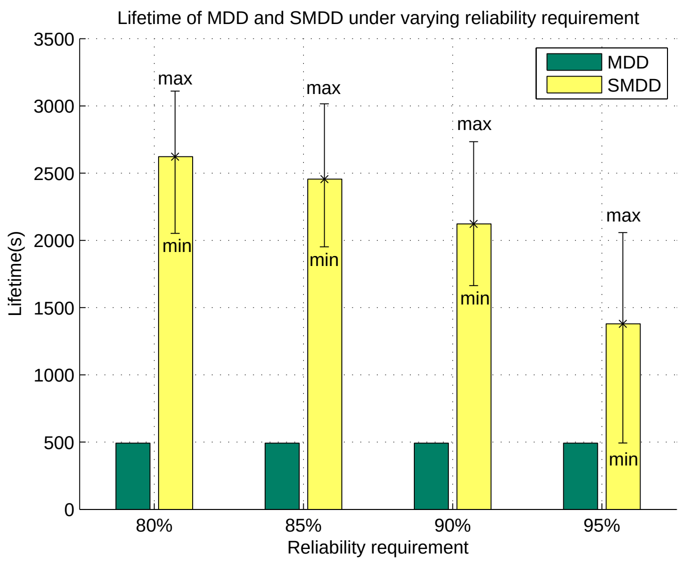


Figure 7.2: Lifetime of Sleeping Multipath Directed Diffusion (SMDD) and no sleeping Multipath Directed Diffusion (MDD) under varying reliability requirements.

7.3.2 Varying the Node Density

In this experiment, we vary the number of nodes in the network from 30 to 70. Fig. 7.3 shows that the lifetime of Sleeping Multipath Directed Diffusion (SMDD) under different reliability requirements increases as the node density increases. This is because higher node density provides higher routing redundancy in the network, and thus more disjoint paths can be discovered to extend the network lifetime. Therefore, Sleeping Multipath Routing can significantly extend the network lifetime, especially in networks with high density.

7.4 Summary

In this chapter, we propose Sleeping Multipath Routing to trade off between reliability and lifetime in wireless sensor networks. The idea of Sleeping Multipath Routing is to use the minimum number of disjoint paths from the source node to the sink node to guarantee data reliability, and put the nodes that are not on the selected paths to sleep, saving energy for later use. Our proposed Sleeping Multipath Routing is a general algorithm, which can be implemented on any multipath routing protocol that discovers mul-

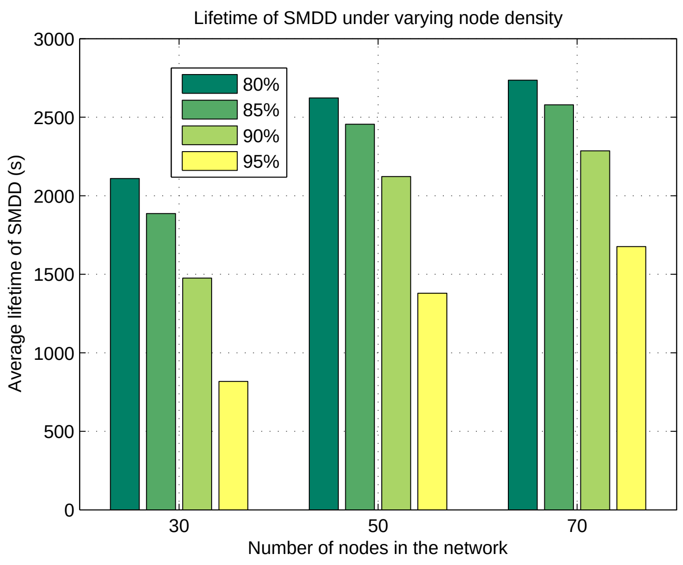


Figure 7.3: Lifetime of Sleeping Multipath Directed Diffusion (SMDD) under varying node density.

tiple disjoint paths between two nodes. We implement Sleeping Multipath Routing on Directed Diffusion, a well-known routing protocol for wireless sensor networks. NS-2 simulation results show that our proposed Sleeping Multipath Routing significantly prolongs the network lifetime, and it can trade off between reliability and lifetime by controlling the number of activated disjoint paths.

Chapter 8

Adaptive Sensor Sleeping

In Chapter 7, we proposed Sleeping Multipath Routing to trade off reliability with network lifetime by putting sensors to sleep at the routing layer. In this chapter, we investigate the performance of multi-layer sensor sleeping using Sleeping Multipath Routing and duty-cycled MAC protocols. Specifically, we simulate a network where Sleeping Multipath Directed Diffusion and S-MAC are used simultaneously to prolong the network lifetime while maintaining a certain reliability to meet the application's requirements.

Simulations in Chapter 3 show that coordination is needed for multi-layer sensor sleeping. Based on the Markov model we have proposed in Chapter 4, we propose an adaptive sensor sleeping solution that coordinates Sleeping Multipath Directed Diffusion and S-MAC to handle dynamic reliability requirements and varying network conditions.

This chapter is organized as follows. Section 8.1 recalls the designs of Sleeping Multipath Directed Diffusion and S-MAC. Sections 8.2, 8.3, and 8.4 elaborate on the coordinations between Sleeping Multipath Directed Diffusion and S-MAC, including information sharing between layers, differentiated data delivery, and timer design. Section 8.5 shows the simulation results of our proposed adaptive sensor sleeping solution under varying application requirements and network conditions. Finally, section 8.6 summarizes the chapter.

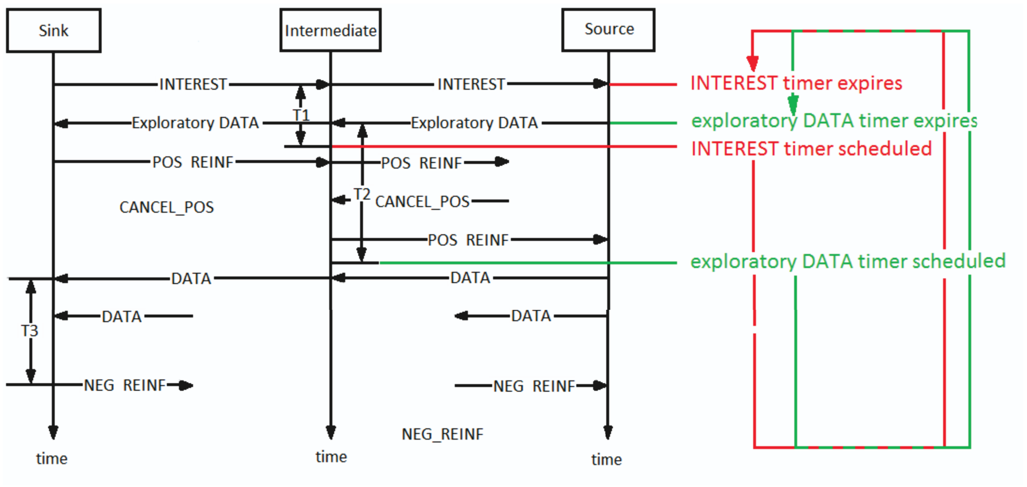


Figure 8.1: Revisit of Sleeping Multipath Directed Diffusion.

8.1 Overview of Sleeping Multipath Directed Diffusion and S-MAC

8.1.1 Sleeping Multipath Directed Diffusion

8.1.1.1 Signaling

Fig. 8.1 shows the operation of Sleeping Multipath Directed Diffusion. There are 2 types of flooding packets, namely INTEREST packets and exploratory DATA packets, and 4 types of unicast packets, namely POS_REINF packets, CANCEL_POS packets, NEG_REINF packets and DATA packets. Specifically, the 2 floodings happen periodically. The INTEREST flooding is initiated by the sink node, and the exploratory DATA flooding is initiated by the source node. The INTEREST flooding and the exploratory DATA flooding may have different periods.

(1) When the first INTEREST flooding reaches a source node, the source node responds with an exploratory DATA flooding. From this point, both INTEREST flooding and exploratory flooding happen periodically.

(2) The sink node unicasts POS_REINF packets to all its neighbors that have delivered exploratory DATA packets to it.

(3) A node that receives a POS_REINF packet forwards the packet to one of its neighbors from which the node received the corresponding exploratory DATA packets.

(4) To guarantee disjoint multipath discovery, a node that receives a POS_REINF packet first checks if it has already been reinforced by another node. To avoid QoS pauses, the node also checks if its remaining lifetime is longer than an exploratory DATA interval. If the node has already been reinforced or its remaining lifetime is shorter than the exploratory DATA interval, the node replies with a CANCEL_POS packet to the sender of the second POS_REINF packet.

(5) A node who receives a CANCEL_POS packet unicasts a POS_REINF packet to another qualified neighbor. If no qualified neighbors are available, the node unicasts a CANCEL_POS packet to the neighbor that initially reinforced the node.

(6) When the source node receives a POS_REINF packet, it unicasts DATA packets periodically along the path from which the POS_REINF packet arrives.

(7) After some time, the sink node receives DATA packets from multiple routing paths.

(8) The sink node determines which paths should be activated to meet the application's reliability requirement according to the remaining lifetime and data reliability of each routing path (which is piggybacked in the DATA packets).

(9) The sink node unicasts NEG_REINF packets along the routing paths that do not need to be activated to support the application's reliability.

(10) Finally, the source node stops sending DATA packets to the neighbors that deliver NEG_REINF packets to it.

8.1.1.2 Timers

As shown in Fig. 8.1, there are three timers introduced in the operation of Sleeping Multipath Directed Diffusion. The first timer is an INTEREST timer, which is scheduled after every INTEREST flooding and expires before the next INTEREST flooding. There is a time gap, defined as T_1, between when the INTEREST timer fires and when it is rescheduled. T_1 to has to be set properly so that during T_1 the node can successfully receive exploratory DATA packets and forward them to the sink node. Every node in the network has an INTEREST timer.

The second timer is an exploratory DATA timer, which is scheduled after every exploratory DATA flooding and expires before the next exploratory DATA flooding. There is a time gap, defined as T_2, between when the exploratory DATA timer fires and when it is rescheduled. T_2 has to be set properly so that during T_2 the node can successfully receive POS_REINF packets (if there is a POS_REINF packet sent to this node), forward the POS_REINF packet to the next hop, or send back a CANCEL_POS packet. Every node in the network has an exploratory DATA timer.

The third timer, a NEG_REINF timer, is at the sink node only. The NEG_REINF timer is scheduled right after the sink node receives the first DATA packets after it starts a positive reinforcement process. The NEG_REINF timer expires in a period of time T_3, and then the sink node sends NEG_REINF packets along the redundant routing paths that can go to sleep. T_3 has to be set properly so that during T_3 the sink node can receive DATA packets from multiple routing paths, and hence select the appropriate number of paths to meet the application's reliability requirement.

8.1.1.3 Sleeping Control

Sleeping Multipath Directed Diffusion puts a sensor to sleep in the following cases: (1) the sensor is not positively reinforced and the sensor's INTEREST timer *and* exploratory DATA timer are pending, or (2) the sensor is negatively reinforced and both the sensor's INTEREST timer *and* exploratory DATA timer are pending.

Meanwhile, Sleeping Multipath Directed Diffusion wakes up a sensor or keeps a sensor awake if: (1) the sensor's INTEREST timer *or* exploratory DATA timer expires, or (2) the sensor is positively reinforced and it has not been negatively reinforced.

8.1.2 S-MAC

As a duty-cycled MAC protocol, S-MAC puts sensors to sleep and wakes them up periodically. A cycle is defined as the interval between two successive sleep periods, or the interval between two successive awake periods. A duty cycle is defined as the length of the awake period in a cycle divided by the cycle length. S-MAC has a fixed awake period in a cycle, hence as the duty cycle changes, the cycle length varies. Meanwhile, S-MAC synchronizes all the nodes in the network so that they sleep and wake up at the same time.

S-MAC requires that every node has one transmission opportunity per cycle. In the awake period of a cycle, S-MAC transmits a packet after a random backoff process. To reduce collisions, S-MAC uses RTS/CTS/DATA/ACK sequences to transmit unicast packets. However, broadcast packets suffer from collisions due to the hidden terminals in the network.

S-MAC uses a FIFO queue to handle incoming packets. Detailed information about S-MAC can be found in Chapter 5 and in [22].

8.2 Information Sharing Between Layers

Sensor sleeping using both Sleeping Multipath Directed Diffusion and S-MAC involves information sharing between four layers: the MAC layer, the routing layer, the transport layer, and the application layer. Specifically, the shared information between the MAC layer and the routing layer includes the sleeping status of a sensor, the remaining lifetime of a routing path, and the data reliability over a routing path. The shared information between the routing layer and the transport layer includes the results of the multipath selection. The shared information between the routing layer and the application layer includes the reliability requirements.

8.2.1 Information Sharing Between the MAC and Routing Layers

There is some information that has to be shared between Sleeping Multipath Directed Diffusion and S-MAC. First of all, the routing layer has a higher priority to put sensors to sleep. Specifically, when the routing layer puts a sensor to sleep, the node is not involved in the data delivery, and hence waking up the sensor periodically at the MAC layer is a waste of energy. Therefore the routing layer should inform the MAC layer of its sensor sleeping decisions. In our case, S-MAC is notified by Sleeping Multipath Directed Diffusion that a sensor is scheduled to sleep at the routing layer, so that S-MAC stops waking up the sensor during the active period of each cycle. Similarly, Sleeping Multipath Directed Diffusion informs S-MAC that a sensor is activated when the sensor's INTEREST timer or exploratory DATA timer expires, i.e., a flooding is going on in the network. As a result, S-MAC continues to wake up the sensor during the active period of each cycle.

Moreover, at the sink node, S-MAC reports the remaining lifetime and data reliability of a routing path to the routing layer, so that the routing layer can do multipath selection accordingly. To achieve this, every DATA packet carries some information of the routing path along which it is transmitted. The information includes (1) the shortest remaining lifetime of a sensor that is on the routing path, and (2) the overall data reliability of the routing path, which equals the product of the reliabilities of every single link of the route. Therefore, when a sensor receives a DATA packet from one of its neighbors, S-MAC checks the remaining lifetime of the sensor, and estimates the data reliability between itself and the neighbor that forwards the DATA packet to it. The data reliability α over a link can be estimated over time T

$$\alpha = N_{suc}/(N_{suc} + N_{err}) \tag{8.1}$$

where N_{suc} is the number of packets that are successfully received by the sensor over this link during T, and N_{err} is the number of packets that are received with errors over this link during T. Since N_{suc} and N_{err} are statistics over the link, the data reliability α can only be estimated at the MAC layer. S-MAC updates the remaining lifetime and the data reliability in the DATA packet, and then forwards the DATA packet to the next hop. When the DATA packet reaches the sink node, it carries the remaining lifetime and the data reliability of the routing path along which the DATA packet travels. S-MAC then reports this information to the routing layer.

8.2.2 Information Sharing Between the Routing and Upper Layers

Information sharing is also necessary between the routing layer and the application layer at the sink node when sensors can be put to sleep by both Sleeping Multipath Directed Diffusion and S-MAC. First of all, the application layer has to inform the routing layer about the requirement on data reliability, so that the routing layer can select routing paths accordingly. The reliability requirement can be a constant value during the lifetime of the network, or it can vary from time to time as needed. Whenever the reliability requirement changes, the application layer informs the routing layer about the change.

On the other hand, the routing layer of the sink node is responsible for informing the transport layer of how many disjoint paths are activated, and what FEC code is used over multiple routing paths to meet the reliability requirements.

8.3 Differentiated Packet Delivery

As shown in Fig. 8.1, Sleeping Multipath Directed Diffusion uses POS_REINF and CANCEL_POS packets to discover multiple disjoint paths in the network during T_2. If no POS_REINF packet successfully reaches the source node during T_2, the source node will not be positively reinforced, and hence no DATA packet will be transmitted to the sink node. Or, if only a few POS_REINF packets are received by the source node during T_2, the sink node may not discover enough disjoint multipaths to meet the application's reliability requirement. Therefore, it is important to guarantee the delivery of POS_REINF and CANCEL_POS packets during T_2 at each node.

However, during T_2, there are other packets that may exist in the network, such as INTEREST packets, exploratory DATA packets, and DATA packets. Some of these packets may be in front of POS_REINF and CANCEL_POS packets in the FIFO queue at a node. In this case, a POS_REINF packet or a CANCEL_POS packet has to wait until all the earlier packets are transmitted, causing a longer delay. As a result, when the routing path between the sink node and the source node includes multihop transmissions, few POS_REINF packets may reach the source node during T_2. Therefore, POS_REINF and CANCEL_POS packets should have higher priority in the data transmissions. One FIFO queue is no longer appropriate for multi-layer sensor sleeping using both Sleeping Multpath Directed Diffusion and S-MAC.

Moreover, NEG_REINF packets should also be transmitted with high priority in order to (1) put sensors to sleep in time to save energy, and (2) to avoid possible confusion between NEG_REINF and POS_REINF if there is a following positive reinforcement.

Therefore, POS_REINF, CANCEL_POS, and NEG_REINF packets should have higher priority than other types of packets in the data transmissions. We propose using 2 FIFO queues to differentiate urgent packets and normal packets in the network, as shown in Fig. 8.2. When a sensor wakes up at the MAC layer, S-MAC first checks if the node has urgent packets to send. If the urgent queue is not empty, the urgent packets will be transmitted in the order that they were received. If the urgent queue is empty, S-MAC transmits normal packets one by one.

Differentiated packet delivery has significant impact on the performance of Sleeping Multipath Directed Diffusion and S-MAC. Without differentiated packet delivery, T_2 has to be set for a much longer time at every node in order to discover multiple paths.

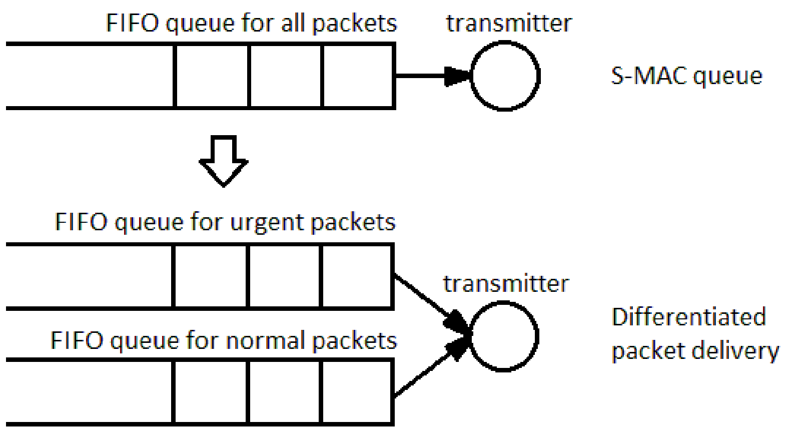

FIFO queue for all packets
transmitter
S-MAC queue
FIFO queue for urgent packets
transmitter
FIFO queue for normal packets
Differentiated
packet delivery

Figure 8.2: S-MAC queue vs. differentiated packet delivery.

Note that nodes have to be awake during T_2, hence a longer T_2 means more energy consumption and shorter network lifetime. Differentiated packet delivery enables a shorter T_2, and hence improves the network lifetime.

8.4 Timer Design

Sleeping Multipath Directed Diffusion has three timers, INTEREST timer, exploratory DATA timer, and NEG_REINF timer. Fig. 8.1 shows T_1, T_2, and T_3, corresponding to the time that the INTEREST timer is idle, the exploratory DATA timer is idle, and the NEG_REINF timer is pending, respectively. During T_1 and T_2, all the nodes in the network have to be awake to deliver flooding packets and to discover disjoint routes. During T_3 all the nodes that are positively reinforced have to be awake to deliver DATA packets, so that the sink node can make decisions accordingly on multipath selection. Therefore, to save energy and prolong the network lifetime, the shorter T_1, T_2, and T_3 are, the better.

On the other hand, T_1, T_2, and T_3 have to be long enough to guarantee the proper functioning of Sleeping Multipath Directed Diffusion. Specifically, T_1 has to be long enough so that the INTEREST packets can travel from the sink node to the source node, and the exploratory DATA packets can be delivered from the source node back to the sink node. T_2 has to be long enough so that the exploratory DATA packets can be flooded from the source node to the sink node, and the source node can be finally reinforced by receiving POS_REINF packets along multiple disjoint paths. T_3 has to be long enough so that the sink node can collect DATA packets from many different disjoint routing paths, and therefore the sink node can select some of the paths to support the application's reliability requirement. If T_3 is too short, the sink node may receive DATA packets from only a few disjoint routing paths, which cannot provide the required reliability.

Therefore, T_1, T_2, and T_3 should be carefully designed considering both energy saving and the proper functioning of the routing protocol. The sink node decides T_1, T_2, T_3, and then it uses the INTEREST flooding to inform all the other nodes in the network about these parameters. A node that receives an INTEREST packet updates the values of T_1, T_2, and T_3 in its S-MAC protocol.

8.4.1 Timer Design According to the Duty Cycle of S-MAC

Chapter 7 uses fixed T_1, T_2, and T_3, for example 5 s, as IEEE 802.11 MAC is not a duty-cycled MAC protocol. However, as the duty cycle changes in S-MAC, the fixed T_1, T_2, and T_3 correspond to different numbers of cycles. For example, if the awake period in a cycle is 28.56 ms, then the cycle lengths for 10% duty cycle, 30% duty cycle, 50% duty cycle, 70% duty cycle, and 90% duty cycle are 285.7 ms, 95.3 ms, 57.2 ms, 40.9 ms, and 31.8 ms, respectively. Therefore, a 5s time interval is approximately 17.5, 52.5, 87.5, 122, and 157 cycles for these duty cycles.

Since S-MAC provides one transmission opportunity to each node per cycle, given a network with a light traffic load (no packet drop due to queue overflow), S-MAC delivers the traffic within the same number of cycles under different duty cycles. Therefore, T_1, T_2, and T_3 should be set to fixed numbers of S-MAC cycles, rather than a fixed time.

8.4.2 Timer Design According to the Markov Model of S-MAC

We proposed a Markov model in Chapter 4 to analyze the throughout, delay, and energy consumption of duty-cycled MAC protocols. Particularly, we applied the Markov model to S-MAC in Chapter 5 under varying network conditions and data arrival rates. Our Markov model has 4 inputs: (1) the number of nodes in the network, (2) the data arrival rate at each node, (3) the queue capacity at each node, and (4) the contention window size at each node. In this chapter, we fix the queue capacity at each node to 10 packets, and we fix the contention window size at each node as 128. Since the Markov model provides the delay of S-MAC over a single link in the network, we can use this model to estimate what are the appropriate number of cycles for T_1, T_2, and T_3. Specifically, we need to determine (1) which Markov model to use (Markov model with or without retransmission, Markov model in single hop or multihop networks), (2) what is the data arrival rate in the network, and (3) how to calculate the packet delay.

8.4.2.1 Timer Design for T_1

First, we determine the appropriate setting of T_1. During T_1, a node floods INTEREST packets and exploratory DATA packets. Since INTEREST packets and exploratory DATA packets are broadcast packets, S-MAC does not use RTS/CTS/DATA/ACK sequences to avoid collisions. Meanwhile, a broadcast packet will not be retransmitted if there is a collision. Although our proposed Markov model is designed for unicast packets, the transmission of a broadcast packet is the same as the transmission of a unicast packet in the case of no retransmissions. Therefore, we use our proposed Markov model for S-MAC with no retransmissions in multihop networks to estimate the delay of broadcast packets during T_1.

During T_1, broadcast packets are flooded throughout the network, hence all the nodes in the network have packets to forward. In the worst case, all the nodes in the network have a packet to send in the same cycle. Then, the situation of S-MAC in this cycle is the same as the situation that S-MAC saturates (all the nodes have a packet to send in every cycle). Therefore, when using the Markov model for S-MAC with no retransmissions to estimate the delay of broadcast packets during T_1, we set the data arrival rate large enough so that S-MAC saturates, in order to obtain a conservative (worst-case) estimate for T_1.

Our proposed Markov model estimates the contending delay and queuing delay of a packet. The contending delay for saturated S-MAC tells how long a broadcast packet can take to be transmitted over 1 hop. However, the queuing delay for saturated S-MAC does not apply to the broadcasts during T_1. This is because broadcast packets are generated rarely (e.g., every 200s) compared to the unicast DATA packets (e.g., 1 packet per 2s), hence most of the time, the network carries light traffic (DATA packets only, on a few reinforced paths), and the average queue length at each node is very small. As a result, it is reasonable to assume that no DATA packets are queued in front of the broadcast packets during T_1. Therefore the queuing delay of a broadcast packet can be neglected. Hence, the average delay of a broadcast packet $avgD_b$ during T_1 can be estimated by the contending delay of a packet in a saturated multihop network with no retransmissions.

Since $avgD_b$ is an estimated average delay of the broadcast packets during T_1, sometimes a broadcast packet may experience a shorter or a longer delay than $avgD_b$. To guarantee the proper functioning of the routing protocol, it is necessary to conservatively estimate the 1-hop delay D_b of a broadcast packet during T_1. We use

$$D_b = \beta \cdot avgD_b \tag{8.2}$$

where $\beta \geq 1$, to estimate D_b using $avgD_b$. Simulations show that $\beta = 2$ provides a good approximation.

Given the network size and the transmission range of a node, we can obtain the number of hops, defined as $hops$, across the network. For example, if a network is deployed in a 100 m $\times$ 100 m area, and each node has a 50 m transmission range, an INTEREST packet takes 2 hops on average to traverse the network, and an exploratory DATA packet takes another 2 hops to travel back to the sink, on average. Therefore, T_1 can be set as 4 times the conservative 1-hop packet delay using our proposed Markov models.

In general,

$$T_1 = 2 \cdot hops \cdot D_b \tag{8.3}$$

Table 8.1 shows the estimates of T_1 using our proposed Markov model for S-MAC with no retransmissions. Nodes are randomly deployed in a 100 m $\times$ 100 m network. There is one sink node and one source node. Packet delay is measured in units of S-MAC cycles.

8.4.2.2 Timer Design for T_2

The design of T_2 is similar to the design of T_1. During T_2, a node floods exploratory DATA packets, and forwards POS_REINF and CANCEL_POS packets, if any. Hence

Table 8.1: Estimates of T_1 When $hops = 2$

$Number of Nodes$	10	20	30	40
$avgD_b(unit : cycle)$	3.9	6.7	9.4	12.1
$D_b(unit : cycle)$	7.8	13.4	18.8	24.2
$T_1(unit : cycle)$	31.2	53.6	75.2	96.8

the conservative 1-hop delay of transmitting an exploratory DATA packet is the same as D_b, the conservative 1-hop delay of transmitting a broadcast packet during T_1.

However, POS_REINF and CANCEL_POS are unicast packets, and hence they will be retransmitted if a collision happens. If we consider retransmitted unicast packets as multiple broadcast packets, our saturated Markov model for S-MAC with no retransmissions can still be used to estimate the average 1-hop delay of unicast packets $avgD_u$ during T_2 [1]. The Markov model with retransmissions cannot be used here because broadcast packets that are also transmitted during T_2 do not fit into the retransmission model.

We define (1) p as the probability for each node to transmit a broadcast packet using the saturated Markov model for S-MAC with no retransmissions, (2) p_s as the probability for each node to successfully transmit a broadcast packet using the saturated Markov model for S-MAC with no retransmissions, and (3) R as the number of retransmissions that a unicast packet can have. The average 1-hop delay of a unicast packet $avgD_u$ during T_2 can thus be obtained as

$$avgD_u = \frac{\sum_{n=1}^{\infty} \sum_{i=0}^{min(n,R-1)} n \cdot p_s \cdot (p - p_s)^i \cdot (1-p)^{n-i}}{\sum_{n=1}^{\infty} \sum_{i=0}^{min(n,R-1)} p_s \cdot (p - p_s)^i \cdot (1-p)^{n-i}} \tag{8.4}$$

As before, we use

$$D_u = \beta \cdot avgD_u \tag{8.5}$$

where $\beta \geq 1$, to conservatively estimate the 1-hop delay of a unicast packet D_u using $avgD_u$. Simulations show that $\beta = 2$ provides a good approximation here as well.

[1]When S-MAC does not saturate, the contention in the network without retransmissions is lower than the contention in the network with retransmissions. Hence the unsaturated Markov model for S-MAC with no retransmissions cannot be used to estimate the delay of a packet that can be retransmitted.

Table 8.2: Estimates of T_2 When $hops = 2$

$NumberofNodes$	10	20	30	40
$avgD_u(unit : cycle)$	4.4	8.2	13.0	18.1
$D_u(unit : cycle)$	8.8	16.4	26.0	36.2
$T_2(unit : cycle)$	33.2	59.6	89.6	120.8

Therefore, T_2 can be estimated as

$$T_2 = hops \cdot (D_b + D_u) \tag{8.6}$$

Table 8.2 shows the estimates of T_2 using our proposed Markov model for S-MAC with no retransmissions, assuming nodes are randomly deployed in a 100 m $\times$ 100 m network. There is one sink node and one source node, and the retransmission limit is $R = 5$. Packet delay is measured in units of S-MAC cycles.

8.4.2.3 Timer Design for T_3

During T_3, a node receives DATA packets and forwards the DATA packets to the sink node along the reinforced routing paths. According to Fig. 8.1, the source node floods exploratory DATA packets periodically. The sink node receives the exploratory DATA packets and then unicasts POS_REINF packets. When the source node receives a POS_REINF packet, it unicasts the DATA packets along the path from which the POS_REINF packet comes.

Therefore, the time interval between an exploratory DATA flooding from the source node to the reception of a DATA packet at the sink node $T_{ex2data}$ is

$$\begin{aligned} T_{ex2data} &= hops \cdot D_b + hops \cdot D_u + hops \cdot D_u \\ &= hops \cdot (D_b + 2 \cdot D_u) \end{aligned} \tag{8.7}$$

In the ideal case, the exploratory DATA packet, the POS_REINF packets, and the DATA packets are successfully delivered without extra delay (every single hop transmission takes 1 cycle). Then the minimum time interval $minT_{ex2data}$ between the exploratory DATA flooding from the source node to the reception of the DATA packet at the sink node is

$$minT_{ex2data} = 3 \cdot hops \tag{8.8}$$

Table 8.3: Estimates of T_3 When $hops = 2$

$Number of Nodes$	10	20	30	40
$D_b(unit : cycle)$	7.8	13.4	18.8	24.2
$D_u(unit : cycle)$	8.8	16.4	26.0	36.2
$T_3(unit : cycle)$	44.8	86.4	135.6	187.2

Since T_3 starts from the time that the sink node receives the first DATA packet after each exploratory DATA flooding, we have

$$\begin{aligned} T_3 &= T_{ex2data} - minT_{ex2data} \\ &= hops \cdot (D_b + 2 \cdot D_u - 3) \end{aligned} \tag{8.9}$$

Table 8.3 shows the estimates of T_3 using our proposed Markov model for S-MAC with no retransmissions, assuming nodes are randomly deployed in a 100 m $\times$ 100 m network. There is one sink node and one source node, and the retransmission limit is $R = 5$. Packet delay is measured in units of S-MAC cycles.

8.5 Adaptive Sensor Sleeping Simulation Results

In this section, we explore how the adaptive sensor sleeping solution handles dynamic application requirements and network conditions using NS-2 simulations.

8.5.1 Varying Reliability Requirements

An application's reliability requirement may change from time to time. For example, a traffic monitoring application may require 95% reliability during rush hour, but 85% reliability throughout the rest of the day. Our proposed adaptive sensor sleeping solution using both Sleeping Multipath Directed Diffusion and S-MAC can adapt to varying reliability requirements by selecting new disjoint multipaths at the sink node whenever the reliability requirement changes.

Specifically, the sink node is informed by the application layer of a new reliability requirement. The sink node then unicasts NEG_REINF packets along all the reinforced routing paths before the next exploratory DATA flooding. When the sink node receives exploratory DATA packets from the source node, it discovers disjoint multipaths by sending POS_REINF packets and receiving DATA packets. At the end of T_3, the sink node decides which paths to keep alive according to the new reliability requirement, and then negatively reinforces the rest of the routing paths after T_3 expires.

Fig. 8.3 shows the network lifetime of adaptive Sleeping Multipath Directed Diffusion and S-MAC, and the network lifetime of non-adaptive Sleeping Multipath Directed Diffusion and S-MAC when the application's reliability requirement varies over time. There are 20 nodes randomly deployed in a 100 m $\times$ 100 m area. There is one sink node and one source node in the network. The sink node and the source node have unlimited energy supplies, while the other nodes in the network have an initial energy of 22 J. The INTEREST flooding period is 150 s, and the exploratory DATA flooding period is 100 s. $T_1 = 53.6$ cycles, $T_2 = 59.6$ cycles, and $T_3 = 86.4$ cycles.

For the non-adaptive Sleeping Multipath Directed Diffusion and S-MAC, no matter how the application's reliability requirement changes, the maximum reliability of 95% is always supported by the network. However, for the adaptive Sleeping Multipath Directed Diffusion and S-MAC, the application's reliability requirement starts from a higher value and changes to 80% after the application runs for 700 s. Fig. 8.3 shows that when the sink node adaptively selects routing paths according to the varying reliability requirement, the network lifetime is much longer than the network lifetime when the sink node is not adaptive to the varying reliability requirement. This is because the the non-adaptive sink node has to always guarantee the highest reliability requirement although more options are available to support the actual lower reliability requirement. Fig. 8.3 also shows that as the duty cycle increases, the network lifetime decreases since node are sleeping for a shorter time.

8.5.2 Changing Network Conditions

Network conditions may also change over time. For example, the number of nodes in the network decreases over time as more and more nodes run out of energy. When the number of nodes in the network is low, more nodes may be added into the network to support the application. The number of nodes in the network influences the timer values of T_1, T_2, and T_3. Therefore, when the number of nodes in the network changes,

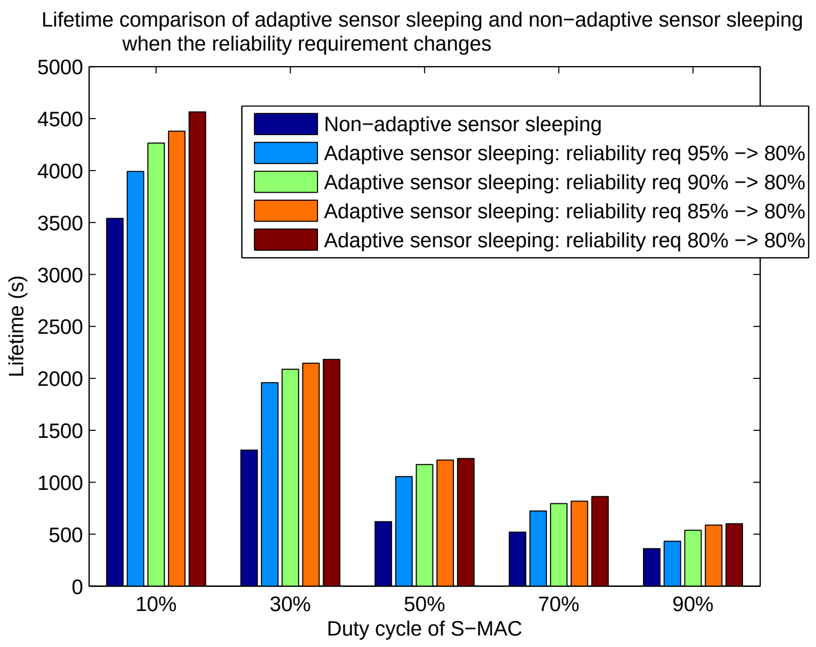


Figure 8.3: Lifetime comparison of adaptive sensor sleeping and non-adaptive sensor sleeping when the reliability requirement changes.

the sink node is responsible for calculating the new T_1, T_2, and T_3, and informing all the other nodes in the network about the changes through INTEREST flooding. Moreover, the sink node should negatively reinforce all the active routing paths before the next exploratory DATA flooding in the network, and start a new multipath selection to include all the nodes in the network.

Fig. 8.4 shows the network lifetime of our adaptive Sleeping Multipath Directed Diffusion and S-MAC compared with the network lifetime of the non-adaptive Sleeping Multipath Directed Diffusion and S-MAC when the number of nodes in the network changes from 20 to 40 after the application runs for 500 s. The non-adaptive sensor sleeping solution keeps the small values of $T_1 = 53.6$ cycles, $T_2 = 59.6$ cycles, and $T_3 = 86.4$ cycles after new nodes join the network. On the other hand, the adaptive sensor sleeping solution updates these timer values to $T_1 = 96.8$ cycles, $T_2 = 120.8$ cycles, and $T_3 = 187.2$ cycles after new nodes joint the network at 500 s.

Simulation results show that the adaptive sensor sleeping solution significantly prolongs the network lifetime compared to the non-adaptive sensor sleeping solution. This is because when the number of nodes in the network increases, the old timer values

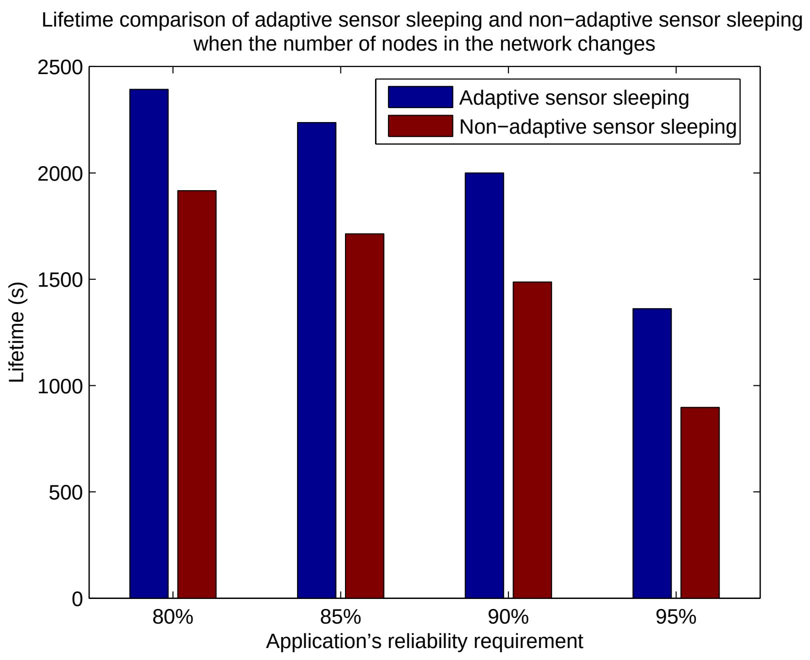


Figure 8.4: Lifetime comparison of adaptive sensor sleeping and non-adaptive sensor sleeping when the number of nodes in the network changes.

become too small for the sink node to discover routing paths. Hence, the newly added routing redundancy cannot be fully utilized to extend the the network lifetime.

8.6 Summary

In this chapter, we propose an adaptive sensor sleeping solution for Sleeping Multipath Directed Diffusion and S-MAC. The adaptive sensor sleeping solution uses Sleeping Multipath Directed Diffusion to handle varying reliability requirements, and uses the Markov model proposed in Chapter 4 to estimate the S-MAC packet delay in order to handle varying network conditions. Simulation results show that our proposed adaptive sensor sleeping solution can dynamically control the key parameters of multi-layer sensor sleeping so as to improve the network lifetime.

Chapter 9

A General Sensor Selection Model to Increase Network Lifetime with QoS Support

The previous chapters discuss different sensor sleeping strategies at the routing and MAC layers. They do not consider application layer sensor sleeping strategies to avoid any application-specific conclusions. However, application layer sensor sleeping can also significantly prolong the network lifetime. It is beneficial to understand how to select sensors according to the application's QoS requirements.

Oftentimes, the application's QoS requirements need to be satisfied by activating the proper sensors in the network. This set of sensors should be rotated over time, both for load balancing and to satisfy dynamic system states and application requirements. This makes it necessary to determine all the possible sensor sets that can support the given QoS requirements for different system states. Since different applications in wireless sensor networks often have different QoS requirements, much past work [62][75][76][77][78] focused on application-specific strategies to satisfy the QoS requirements and prolong the network lifetime. This makes it difficult to accommodate new applications. Therefore we propose a general 4-layer QoS description model to abstract the application QoS requirements and simplify the process of finding all the possible sensor sets that meet the required QoS.

On the other hand, prolonging the network lifetime with QoS support is of great importance, as the sensors are usually battery-powered, and hence highly constrained in terms of their available energy. Previous sensor selection schemes [63][79][80][81][82] are not optimal in terms of extending the network lifetime, as the cost functions they rely on do not simultaneously consider the diversity of various sensors in the network in terms of power consumption, remaining energy and contribution to the application QoS, and hence they cannot reflect a comprehensive estimate of a sensor's value to the application.

Therefore, using our proposed 4-layer QoS description model, we propose a novel cost function to determine a sensor's cost, which can be used for sensor selection [31]. Our cost function, Sensor Usage Index (SUI), is defined as a sensor's relative ideal lifetime divided by its actual remaining lifetime. For a given application, each sensor's relative ideal lifetime is evaluated based on the knowledge of all the possible sensor sets that support the application QoS requirements in various system states. A sensor's relative ideal lifetime reflects how long the sensor should be alive in order to achieve maximum network lifetime. A sensor's SUI compares its relative ideal lifetime with its actual remaining lifetime, and thus SUI provides an idea of how much the sensor is overused. Sensors that have a low SUI can be used more liberally than sensors that have a high SUI. When selecting the final sensor set, high cost sensors will be avoided, if possible. Simulation results show that using SUI for sensor selection outperforms previous cost functions in both homogeneous networks and heterogeneous networks. In fact, networks that perform sensor selection based on SUI have lifetimes that approach the optimal network lifetime.

In this chapter we consider single-hop centralized wireless sensor networks. Wireless senor networks have been gaining increasing prominence for practical deployments [83][84][85], especially for applications that utilize single-hop centralized networks [59][60][61] such as avalanche rescue [86], on-body health monitoring [66], environmental monitoring [67], and others. Several authors have shown that single-hop centralized networks can be more energy-efficient than their multi-hop counterparts [68][87][88][89]. Meanwhile, single-hop centralized networks are cost-effective, easy to deploy and efficient in terms of scheduling data transmissions.

The rest of this chapter is organized as follows. Section 9.1 describes the proposed QoS description model and our novel cost function, SUI, in detail. Section 9.2 analyzes our simulation results, and section 9.3 summarizes the chapter.

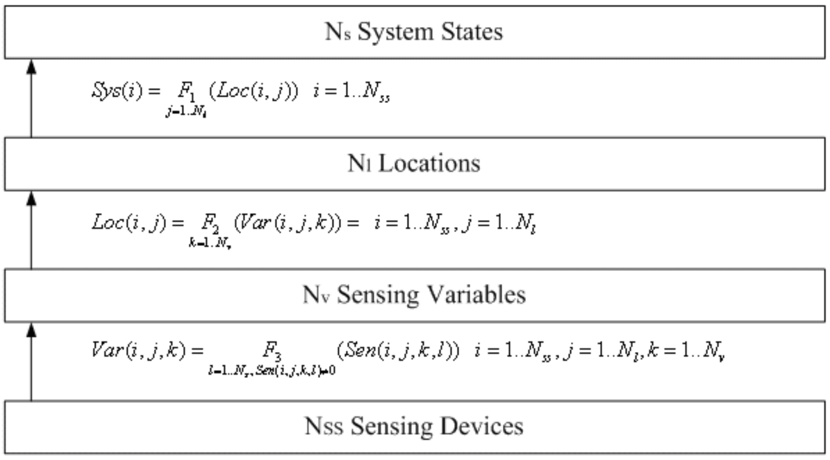

Ns System States
$Sys(i) = \underset{j=1..N_l}{F_1}(Loc(i,j)) \quad i = 1..N_{ss}$
Nl Locations
$Loc(i,j) = \underset{k=1..N_v}{F_2}(Var(i,j,k)) = \quad i = 1..N_{ss}, j = 1..N_l$
Nv Sensing Variables
$Var(i,j,k) = \underset{l=1..N_v, Sen(i,j,k,l)\neq 0}{F_3}(Sen(i,j,k,l)) \quad i = 1..N_{ss}, j = 1..N_l, k = 1..N_v$
Nss Sensing Devices

Figure 9.1: 4-layer QoS description model.

9.1 QoS Description Model and Cost Function

The proposed general sensor selection model consists of two tiers. The first tier is to determine all possible sensor sets to support application QoS in different scenarios. We propose a general QoS description model to achieve this goal with low computational complexity. Other methods that determine appropriate sensor sets that support application QoS could also be used as the first tier. Once these sensor sets are determined, they are used as input for the second tier, which is a sensor selection scheme that employs a novel cost function called SUI. SUI is calculated based on the knowledge of all possible sensor sets (provided by the first tier) to prolong the network lifetime with QoS support. First we discuss our approach for finding the sensor sets (tier 1) using a 4-layer QoS description model, and then we discuss our approach for sensor selection with QoS support (tier 2) using SUI.

9.1.1 4-Layer QoS Description Model

For a given application, a 4-layer model, as shown in Fig. 9.1, is established to describe the application's QoS requirements and the sensor's capabilities. We first introduce the definition of each layer, and then we formulize the QoS relationship between each layer.

The highest layer is the system state layer. An application runs in different system states with a certain probability. In each system state, the application has different QoS requirements. We define the QoS of a system state i as a Boolean variable $Sys(i)$, with logic value 0 meaning "QoS requirements are not satisfied" and logic value 1 meaning "QoS requirements are satisfied". For example, consider a warehouse monitoring application, shown in Fig. 9.2, that has two system states, "open" and "closed", with a probability of occurrence at 0.7 and 0.3, respectively. Thus the system spends about 70% of the time in the "open" state (warehouse is open) and about 30% of the time in the "closed" state (warehouse is closed).

The second layer defines every location of interest j under a particular system state i. Different system states may be interested in different locations, and different locations may have different QoS requirements under the different system states. The QoS of location j under system i is defined as a Boolean variable $Loc(i, j)$, with logic value 0 meaning "QoS requirements are not satisfied" and logic value 1 meaning "QoS requirements are satisfied". Continuing the previous example, suppose that the warehouse monitoring application supervises three locations: door, window and goods. In the "closed" state, either monitoring the door and the window together or monitoring the goods are enough to meet the application QoS requirements, while in the "open" state, it is necessary to monitor the door, the window and the goods simultaneously to meet the application QoS requirements.

The third layer consists of all sensing variables of interest. Different locations under different system states may be interested in different sensing variables. The QoS of a sensing variable k in location j under system state i is defined as a Boolean variable $Var(i, j, k)$, with logic value 0 meaning "QoS requirements are not satisfied" and logic value 1 meaning "QoS requirements are satisfied". For instance, the warehouse monitoring application may utilize sensors for sensing sound and motion around the door and the window, and for sensing motion and vibration around the goods, under both "open" and "closed" system states.

The final layer consists of the sensing devices. Each sensing device has different sensing capabilities, enabling it to sense different variables of interest. The QoS of sensing device l to support sensing variable k in location j under system state i is defined as a Boolean variable $Sen(i, j, k, l)$, with logic value 0 meaning "the sensor does not have the sensing capability" and logic value 1 meaning "the sensor has the sensing capability". For example, in the warehouse monitoring application, one sensor

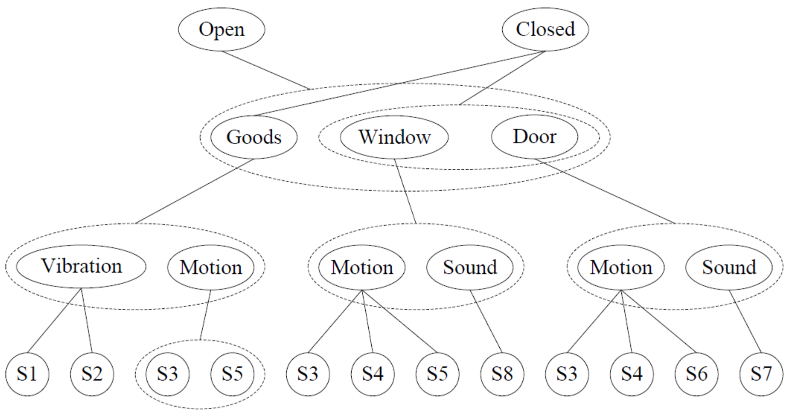


Figure 9.2: QoS description of the warehouse monitoring application.

may be required for each sensing variable at each location, except that at least two motion sensors are required to guarantee the QoS of the goods, under both "open" and "closed" system states.

In the 4-layer QoS description model, each system state, location and sensing variable has its own interest only to the elements in the next lower layer. This interest is actually the QoS requirements between adjacent layers. Hence, the 4-layer QoS description model decouples the application QoS requirements into three levels of independent QoS requirements, between system states and locations, locations and sensing variables, and sensing variables and sensing devices. As we defined the QoS of each element in each layer, the QoS requirements between adjacent layers can be described as a logic function, with the lower layer QoS as input and higher layer QoS as output. Assume there are N_s system states, N_l locations, N_v sensing variables, and N_{ss} sensing devices. The three levels of QoS requirements can be defined as follows.

$$Sys(i) = F_1(Loc(i,j)), i = 1..N_s, j = 1..N_l \tag{9.1}$$

$$Loc(i,j) = F_2(Var(i,j,k)), j = 1..N_l, k = 1..N_v \tag{9.2}$$

$$Var(i,j,k) = F_3(Sen(i,j,k,l)), k = 1..N_v, l = 1..N_{ss} \tag{9.3}$$

where F_1, F_2, and F_3 are logic functions to describe the QoS requirements between adjacent layers, in the form of either sum-of-products or products-of-sums.

These logic functions can be obtained by the application QoS requirements, explicitly or implicitly with low computational complexity. For example, the warehouse monitoring application explicitly requires the interest of each system state, location, and sensing variable. In the case of implicit QoS requirements, such as "locations cover 80% of the given area", "any 2 out of 5 sensing variables", "at least 3 sensing devices available", etc., the logic functions can be obtained by exhaustive search. However, instead of using an overall exhaustive search, our layered exhaustive search has the potential to reduce computational complexity due to the following reasons.

First, the exhaustive search for F_1 and F_2 only depends on the number of locations N_l and the number of variables N_v, respectively. Since N_l and N_v are independent from the total number of sensors N_{ss}, the computational complexity of exhaustive search for F_1 and F_2 are $O(1)$, and thus can be neglected. The exhaustive search for F_3 is related to the number of sensors N_{ss} in the network. However, the number of sensors included in the exhaustive search for each F_3 is oftentimes much smaller than N_{ss} due to the constraints of locations and variables. Given a system state i that includes N_l locations, and each location includes N_v variables, there are $N_l \cdot N_v$ different F_3 searches, each of which includes a subset of N_{ss} sensors that cover the corresponding location and sense the corresponding variable. If each sensor can only support one variable at one location, the sensors included in each F_3 do not overlap. Assuming that the number of sensors included in each F_3 search is equal, then each F_3 search includes $N_{ss}/(N_l \cdot N_v)$ sensors. Therefore, the computational complexity for each F_3 search is $O((N_{ss}/(N_l \cdot N_v))!)$, and the computationally complexity for our proposed 4-layer QoS description model is $O((N_{ss}/(N_l \cdot N_v))!)$. Compared to the overall exhaustive search, which has a computational complexity of $O(N_{ss}!)$, our 4-layer QoS description model greatly reduces the computational complexity.

However, in our 4-layer QoS description model, if each sensor can support multiple variables at multiple locations, the sensors included in each F_3 search may overlap. When all N_{ss} nodes are included in one F_3 search, the computational complexity of our 4-layer QoS description model becomes the same as the computational complexity of the overall exhaustive search.

Additionally, the layered logic functions may incur much less computational complexity than an overall exhaustive search when a sensor joins the network. In the proposed 4-layer QoS description model, updates are only required for the F_3 logic functions that the new sensor supports, while the rest of the logic functions remain unchanged. As a result, re-computation is avoided as much as possible, and the incurred computational complexity is reduced to its minimum (an update of the minimization process of all logic functions is needed to obtain all possible sensor sets, as described later). However, an overall exhaustive search cannot distinguish the influence of a new sensor, and hence has to perform all the computation again.

Moreover, the layered logic function describes QoS requirements with great flexibility. First of all, it supports different QoS requirements under different system states, locations and sensing variables. Hence, it can be easily applied to sophisticated application QoS requirements, simplifying them using the general, layered structure. Second, it also supports applications where sensing accuracy and link quality (measured by the base station) are not ideal. For example, assume each sensor has its own sensing accuracy (e.g., represented by a real number between 0 and 1) to support a given variable. The given variable requires a minimum sensing accuracy, which can be supported by single or multiple sensors whose total sensing accuracy is greater than the threshold. Hence, when doing the exhaustive search for the corresponding logic function F_3, this minimum requirement on sensing accuracy is taken into account. Similarly, QoS requirements on link quality can also be handled in this way.

When all the logic functions are obtained, all possible sensor sets that can support the QoS requirements under each system state can be determined by simply combining and simplifying those logic functions into a summation of products, using Boolean equation minimization or a Karnaugh Map [65]. The minimized logic function is true when any one of the products is true. Correspondingly, the QoS requirements of a system state can be satisfied by any of the possible sensor sets, which are represented by the product terms in the minimized logic function. This means that all the sensors in that sensor set have to be activated, or have a "true" logic value, simultaneously to support the QoS. Note that for each system state, the combined logic function only has one type of redundancy, which can be easily minimized. The redundancy is in the form $F = ABC + AB = AB$. This means that if sensors A and B can support the QoS, sensor C does not need to be activated.

For the warehouse monitoring example, suppose there are 8 sensing devices, named $s_1 s_8$, in which s_1 and s_2 are vibration sensors monitoring the goods, s_3, s_4, s_5 and s_6 are motion sensors covering door/window/goods, door/window, window/goods, and door, respectively, and s_7 and s_8 are sound sensors covering door and window, respectively. To save space, we use the straightforward names "open", "closed", "door", "window", "goods", "vibration", "motion", and "sound" to represent the QoS of the system states, locations and sensing variables. The QoS relationship, shown in Fig. 9.2, can therefore be described as follows:

$$\begin{aligned} open &= door \cdot window \cdot goods \\ &= (sound_{door} \cdot motion_{door}) \\ &\cdot\ (sound_{window} \cdot motion_{window}) \\ &\cdot\ (motion_{goods} \cdot vibration_{goods}) \\ &= [s_7 \cdot (s_3 + s_4 + s_6)] \cdot [s_8 \cdot (s_3 + s_4 + s_5)] \cdot [(s_3 \cdot s_5) \cdot (s_1 + s_2)] \\ &= s_1 s_3 s_5 s_7 s_8 + s_2 s_3 s_5 s_7 s_8 \end{aligned} \tag{9.4}$$

$$\begin{aligned} closed &= door \cdot window + goods \\ &= (sound_{door} \cdot motion_{door}) \\ &\cdot\ (sound_{window} \cdot motion_{window}) \\ &+ (motion_{goods} \cdot vibration_{goods}) \\ &= [s_7 \cdot (s_3 + s_4 + s_6)] \cdot [s_8 \cdot (s_3 + s_4 + s_5)] + [(s_3 + s_5) \cdot (s_1 + s_2)] \\ &= s_3 s_7 s_8 + s_4 s_7 s_8 + s_5 s_6 s_7 s_8 + s_1 s_3 s_5 + s_2 s_3 s_5 \end{aligned} \tag{9.5}$$

Hence, there are 2 possible sensor sets that can support the QoS requirements in the "open" state and 5 possible sensor sets that can support the QoS requirements in the "closed" state. The sensor sets are listed as products in the minimized logic function of "open" and "closed" shown above.

The 4-layer QoS description model provides a general way to abstract the application QoS requirements and determine all possible sensor sets to support the application QoS requirements. In most cases of single-hop centralized networks, it greatly reduces the computational complexity, compared to a blind exhaustive search. Since the main focus of this chapter is to optimize the sensor selection scheme, we take the 4-layer QoS description model as a supportive design to obtain all possible sensor sets for the sensor selection scheme to use. Any other method or algorithm that can determine all possible sensor sets to support the application QoS requirements can be used in place of our 4-layer QoS description model in the sensor selection process, and this will not influence the final conclusion. Hence in the rest of the chapter we will only focus on the design, analysis and simulation of our proposed sensor selection scheme (tier 2).

9.1.2 Sensor Selection Scheme

With the knowledge of all possible sensor sets that meet QoS requirements under each system state, the base station must select a final sensor set from among the candidates to provide data for the current system state while trying to maximize the network lifetime. Our proposed sensor selection scheme first introduces a novel cost function, SUI, which describes how much the sensor is overused compared to its relative ideal lifetime, and then establishes a set of criteria based on the cost function to select the most preferable sensor set, aiming at achieving longer network lifetime.

9.1.2.1 SUI Cost Function

To maximize the network lifetime, we want to use the sensors ideally, so that every sensor can contribute all its energy in supporting the application QoS. To achieve this goal, both energy constraints and QoS requirements need to be considered. First of all, consider the scenario where there are no energy constraints. In this case, the base station can randomly select one sensor set from all possible sensor sets to meet the application QoS. Hence, over time, when the base station has made numerous random selections from the possible sensor sets, each sensor set will support the application for approximately the same amount of time. Consequently, the more often a sensor appears in the possible sensor sets, the longer it tends to be used to support the application QoS. However, if we consider energy constraints, sensors cannot be alive forever. The lifetime of a sensor is determined by its initial energy and its power consumption, and hence the sensor may not be able to support the application in the same way as it did in the scenario of no energy constraints.

Suppose we were able to “assign” a fixed amount of energy arbitrarily to each sensor in the network. Each sensor should receive an amount of energy proportional to the relative amount it required in the scenario with no energy constraints. This will ensure that the network can support application QoS for the entire time it is operational, and at the end of the network lifetime, all the energy in the network (the energy of each individual sensor) will be used, and no energy will be wasted. We call this the "ideal" scenario, and we call a sensorfs lifetime in this scenario its "ideal lifetime".

In reality, however, we cannot assign energy to the sensors. They have a fixed initial energy and a fixed power consumption, which determines their active lifetime. Thus we need to determine how long the sensorfs actual lifetime is relative to its "ideal lifetime". To do this, first we define a sensorfs relative ideal lifetime (SRIL), and then we compare this relative ideal lifetime with the sensor's actual remaining lifetime.

To find a sensors SRIL, we must determine the number of sets in which the sensor is active, and the probability of needing to use those sets. Suppose an application runs in Ns system states, each of which has corresponding QoS requirements and a probability of occurrence $P(i)$, $i = 1..N_s$. For each system state i, there are $N_p(i)$ possible sensor sets (determined in tier 1), described as $F(i,j)$, $i = 1..N_s$, $j = 1..N_p(i)$, that can provide the required QoS. Then, for each sensor k,

$$SRIL(k) = \sum_{1=1..N_s} (P(i) \cdot \sum_{k \in F(i,j):j=1..N_p(i)} \frac{1}{N_p(i)}) \tag{9.6}$$

This definition shows three rules to calculate a sensor's contribution to the application QoS. First, sensors in a given sensor set are supposed to work simultaneously for the same amount of time when that sensor set is chosen to support the applicationfs QoS. Second, since all possible sensor sets under a specific system state equally satisfy the applicationfs QoS, they are assumed to ideally support the system state for the same amount of time (note, however, that some sensor sets may actually be chosen more often according to our proposed cost function, which utilizes SRIL). Third, as described by the 4-layer QoS description model, each system state has a certain probability of occurrence, which also influences a sensorfs contribution in the ideal case. Take s_7 in the warehouse monitoring application for example. s_7 appears in 2 out of 2 possible sensor sets in the "open" state, which has the probability of occurrence of 0.7, and 3 out of 5 possible sensor sets in the "closed" state, which has the probability of occurrence of 0.3. Hence,

$$SRIL(7) = 0.7 \cdot (\frac{1}{2} + \frac{1}{2}) + 0.3 \cdot (\frac{1}{5} + \frac{1}{5} + \frac{1}{5}) = 0.88 \tag{9.7}$$

This mean s_7 has to be active during 70% of the network lifetime to support the "open" state, and be active during 0.18% of the network lifetime to support the "closed" state.

Set as a reference to ideally use the sensors and maximize the network lifetime, a sensors SRIL is compared to its remaining lifetime (SRL) as a cost function, called sensor usage index (SUI) to reflect the extent to which the sensor is being used.

$$SUI(k) = SRIL(k)/SRL(k) \tag{9.8}$$

In the ideal case, every sensor has the same SUI, so that the sensors can contribute all of their energy to the application and maximize the network lifetime. In the practical case, however, sensors have non-ideal lifetime, and consequently are diverse in their SUIs. The sensor with the highest SUI in the network is the bottleneck to prolonging the network lifetime. The lower the SUI, the less the sensor is being used compared to its ideal usage. Hence, sensors with lower SUI are preferred to be used early, as their remaining lifetimes are longer than the bottleneck sensor and hence can support extra network lifetime until the bottleneck sensor has to be used. However, as sensors remaining lifetimes become shorter and shorter, new bottlenecks will arise. Finally, all the sensors keep being used at the same pace, so that they are utilized to their maximum limit, thereby prolonging the network lifetime.

Note that the knowledge of each system state's probability of occurrence is used in the calculation of SUI. However, these probabilities may not be accurately evaluated before starting the application, or sometimes the probabilities may vary over time. We therefore evaluate these probabilities of occurrence on the fly by adaptively updating the values according to the current system state and past experience. The initial probabilities of occurrence can be set as rough evaluations or arbitrary values. Assume T_i, $i = 1..N_s$ represents the time for which the application has been running in system state i. The probability of occurrence of system state i is updated as follows:

$$P(i)_{current} = 0.8 \cdot P(i)_{past} + \frac{0.2 \cdot T_i}{\sum\limits_{j=1}^{N} T_j} \tag{9.9}$$

9.1.2.2 Choosing the Optimal Sensor Set

Since a sensor set usually includes multiple sensors, each of which might have different SUI values, a set of criteria is needed to evaluate every possible sensor set based on the SUI of the individual sensors, so that the most preferable set is selected to prolong the network lifetime.

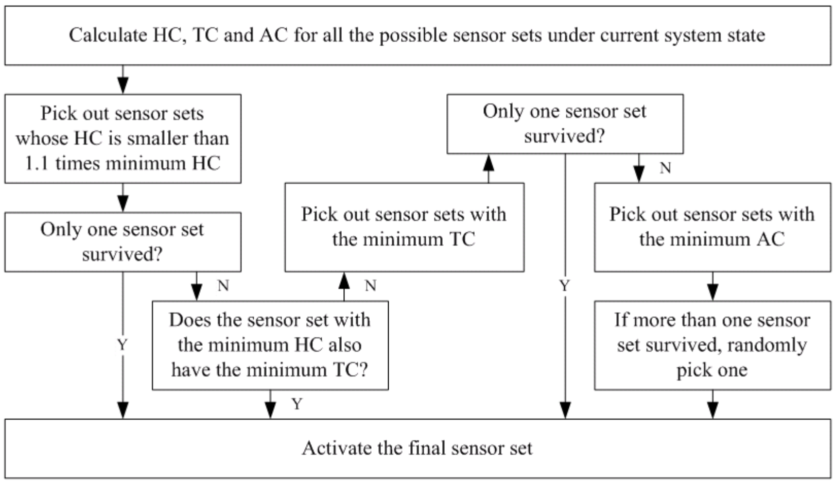

Calculate HC, TC and AC for all the possible sensor sets under current system state
Pick out sensor sets whose HC is smaller than 1.1 times minimum HC
Only one sensor set survived?
Y
N
Does the sensor set with the minimum HC also have the minimum TC?
Y
N
Pick out sensor sets with the minimum TC
Only one sensor set survived?
Y
N
Pick out sensor sets with the minimum AC
If more than one sensor set survived, randomly pick one
Activate the final sensor set

Figure 9.3: Sensor set selection.

In each possible sensor set, the sensor with the highest cost, referred to as HC, determines the remaining lifetime of that sensor set. Thus, under a given system state, the sensor set with the lowest HC is preferred to be selected. This is the first selection criterion. The total cost of a sensor set, referred to as TC, tells how many resources are going to be used at one time. If two sensor sets have the same HC, the one with the smaller TC usually involves fewer sensors, and thus is preferred to be selected. However, if the HC of sensor set A is marginally smaller than the HC of sensor set B, but sensor set A involves a large number of sensors leading to a large TC, while sensor set B uses only one sensor, thus having a very small TC, sensor set B is more preferable since a large gain on TC more than compensates for the tiny loss on HC in terms of prolonging the network lifetime. Hence, it is necessary to loosen the constraint on the lowest HC criterion. Thus, sensor sets with lower HC can also enter the second round selection based on TC. Finally, if two sensor sets have the same HC and TC, smaller average cost within the sensor set, referred to as AC, implies that all the sensors in that set have been used to a similar extent, and less energy will be wasted in the end. Thus, the sensor set with the lowest AC will be finally selected. Fig. 9.3 shows the process of selecting sensor sets based on the individual sensor's cost.

9.1.3 Sensor Selection Process

A base station is the controller in a single-hop centralized wireless sensor network. It is responsible for not only gathering the sensed data from all the sensors, but also establishing and maintaining the 4-layer QoS description model (if it is used), selecting and activating the final sensor set accordingly, and scheduling the transmission of the selected sensors data. The base station knows the applications QoS requirements for different system states, locations and sensing variables before the application begins. The base station builds the 4-layer QoS description model as sensors report their locations, sensing capabilities, initial energies and power consumption parameters when joining the network. Hence, the base station can select the preferable sensor set and then activate the sensors in this set.

We assume that the base station periodically broadcasts a beacon to all the sensors in the network. The interval between successive beacons is called a superframe. A beacon includes information about the activated sensors, and in which order they should transmit data during the current superframe. A beacon may also indicate a contention period following itself and before the data transmission, for any new sensors to join the network. After the beacon, all the activated sensors start their transmissions according to the schedule in a TDMA manner until the next beacon arrives. Note that the 4-layer QoS description model is updated only when a sensor joins the network or dies. In the case of a new sensor joining the network, updates are only required for the F3 logic functions that the new sensor supports. An update of the minimization process of all logic functions is needed to obtain all possible sensor sets. In the case of a dying sensor, the possible sensor sets are updated by simply removing the sets that include the dead sensor. Generally, the base station determines the death of a sensor by either predicting the sensors remaining lifetime based on its own knowledge or parsing piggy-backed information from the dying sensor. Fig. 9.4 illustrates the entire process of sensor selection at the base station.

9.2 Simulation Results

In this section, we use MATLAB simulations to compare our proposed sensor selection scheme using SUI with random selection, minimum power selection and $1/E$ selec-

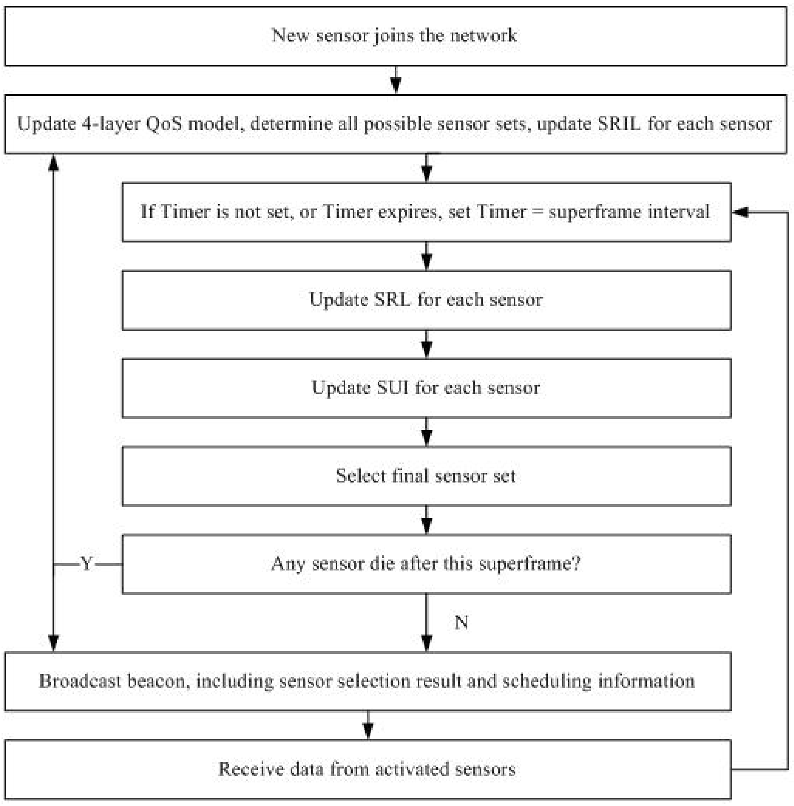

New sensor joins the network
Update 4-layer QoS model, determine all possible sensor sets, update SRIL for each sensor
If Timer is not set, or Timer expires, set Timer = superframe interval
Update SRL for each sensor
Update SUI for each sensor
Select final sensor set
Any sensor die after this superframe?
Y
N
Broadcast beacon, including sensor selection result and scheduling information
Receive data from activated sensors

Figure 9.4: Sensor selection process at the base station.

Table 9.1: CC2420 Transceiver Parameters [87]

Mode	Power (mW)	Mode	Power (mW)

Transmit	25.2($\leq$18m)
	29.7($\leq$56m)
	33.0($\leq$100m)
		Receive	56.4

	42.0($\leq$177m)
	52.5($\leq$314m)
		Sleep	1.0

tion, in terms of network lifetime with QoS support. First, a special case, the warehouse monitoring system described in section 9.1.1, is examined to intuitively show the advantages of our proposed scheme. Then more general cases are studied in homogeneous and heterogeneous sensor networks. We also examine the difference between the network lifetime supported by our proposed sensor selection scheme and the optimal network lifetime. In the simulations, system states are generated randomly according to their probabilities of occurrence, in units of 3 hours. The base station updates the activated sensor set and broadcasts a beacon every half an hour. Activated sensors transmit data every 5 seconds. For each superframe, we assume a 100-byte beacon. For each data transmission, a 6-byte MAC header is included. We use different sensors with the same transceiver, CC2420 from Texas Instruments, an IEEE 802.15.4 radio. The power consumption parameters of the transceiver are listed in Table 9.1.

9.2.1 Case Study: Warehouse Monitoring

In this section, we examine the warehouse monitoring application, as discussed in section 9.1.1. In this application, there are 3 types of sensors: vibration sensors, motion sensors and sound sensors. Their parameters are listed in Table 9.2. Note that the sound sensors power consumption is much lower than that of the other two sensors.

Fig. 9.5 shows the average improvement in terms of network lifetime provided by our proposed sensor selection scheme that utilizes SUI compared with random selection, minimum power selection and $1/E$ selection. Averaged over 1000 random scenarios of the system state sequences, the proposed scheme outperforms the three existing schemes with 26.9%, 21.3% and 27.7% longer lifetime, respectively. Note that $1/E$ selection in this case has similar performance with random selection because $1/E$ se-

Table 9.2: Sensors in Warehouse Monitoring System [77][78][79]

Sensors	Sensing Power (mW)	Battery Capacity (mAH)	Voltage (V)	Sensing Radius (m)	Data Rate (bps)
Vibration	30	1200	9	3	24
Motion	70	2000	3	4	240
Sound	2.25	400	4.5	2	16

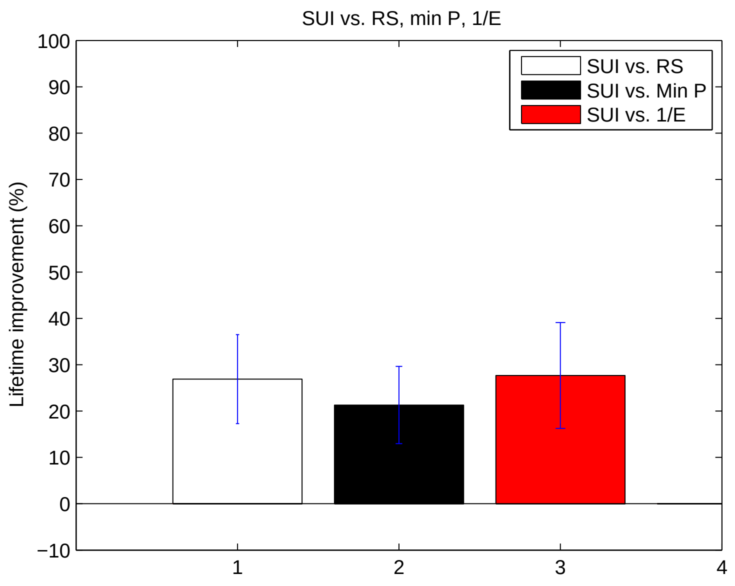


Figure 9.5: Network lifetime comparisons for warehouse monitoring.

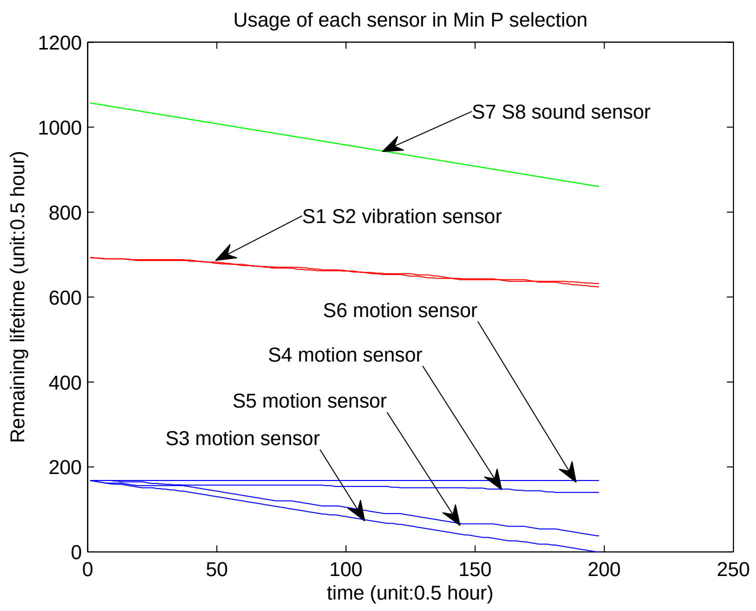


Figure 9.6: Use of sensors in Min P selection.

lection only works well in the homogeneous scenario where all sensors have the same power consumption. Fig. 9.5 also illustrates the standard deviations of the improvements, which are small, illustrating that the proposed sensor selection scheme can provide constant improvement compared to the existing schemes.

To explore the reasons for the longer lifetimes, Fig. 9.6, Fig. 9.7 and Fig. 9.8 show the use of sensors in minimum power selection, 1/E selection and the proposed SUI selection schemes, respectively. For all of the cases, the applicationfs QoS is no longer supported when some of the motion sensors die. Obviously, the motion sensors are the bottleneck in prolonging the network lifetime, since they play an indispensable role in both the “open” and “closed” states, and unfortunately have the shortest initial lifetime (battery capacity divided by power consumption). Observe the possible sensor sets in the “open” state:

$$open = s_1s_3s_5s_7s_8 + s_2s_3s_5s_7s_8 \tag{9.10}$$

s_3 and s_5 are always used together to support the QoS requirements. However, in the “closed” state:

$$closed = door \cdot window + goods \tag{9.11}$$

$$door \cdot window = s_3s_7s_8 + s_4s_7s_8 + s_5s_6s_7s_8 \tag{9.12}$$

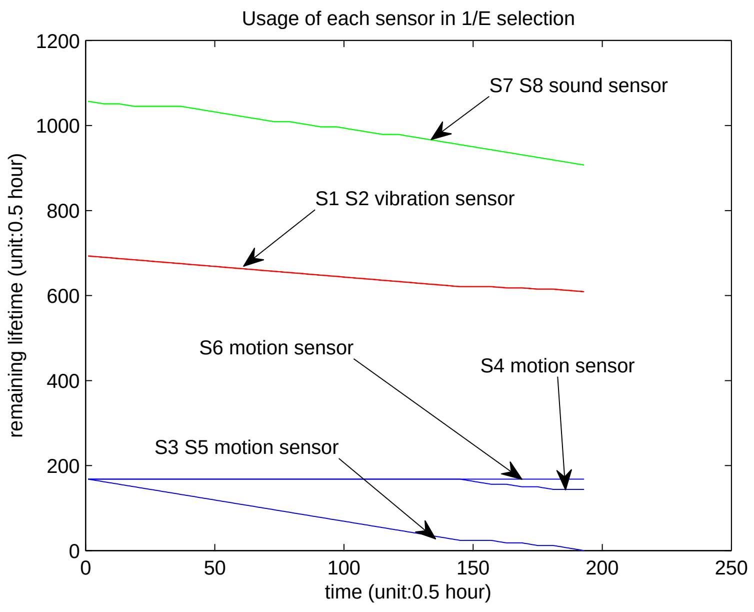


Figure 9.7: Use of sensors in $1/E$ selection.

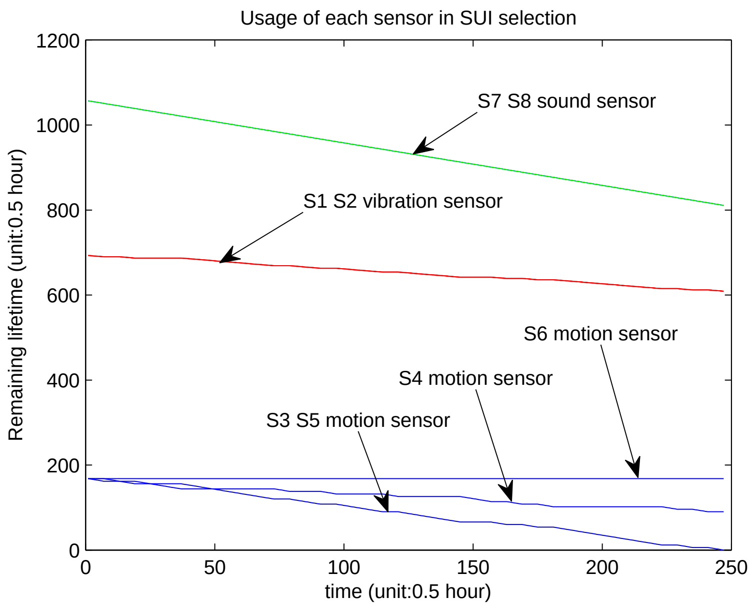


Figure 9.8: Use of sensors in SUI selection.

$$goods = s_1 s_3 s_5 + s_2 s_3 s_5 \tag{9.13}$$

s_3 and s_5 are used simultaneously only in monitoring goods. In the minimum power selection scheme, $s_3s_7s_8$ and $s_4s_7s_8$ are most preferable to be used in the "closed" state due to their minimum power consumption. Hence, these two sensor sets are randomly chosen to support the "closed" state, while s_3 and s_5 are used to support the "open" state. As a result, s_3 has to keep active in all the "open" states and half of the "closed" states, and thus dies quickly. Once s_3 dies, the QoS requirements in the "open" state can no longer be satisfied, in other words, s_3's lifetime determines the network lifetime, even though s_4 and s_6 still have much remaining energy. On the other hand, $1/E$ selection favors monitoring goods, and hence it uses s_3 and s_5 in the "closed" state in the first 70% of the network lifetime, as shown in Fig. 9.7. Since the sound sensors (s_7 and s_8) have much lower initial energy than the other two types of sensors, they contribute large costs to the possible sensor sets that monitor the door and the window. Sensor set $s_4s_7s_8$ is finally used at the end of the network lifetime, as the remaining energy of s_3 and s_5 are low, leading to an even larger cost. Similar to the minimum power selection, $1/E$ selection does not properly utilize s_4 and s_6 in the "closed" state, so that the network lifetime is not optimized. Instead of rarely using s4, SUI selection uses s_4 all the time to support the "closed" state. Since s_4 appears in all the possible sensor sets only once, it has much smaller SRIL than s_3 and s_5, but their initial lifetimes are the same. Hence, s_4 has much smaller SUI than s_3 and s_5, making the sensor set $s_4s_7s_8$ have the minimum HC among all the candidates. Therefore, s_3 and s_5 contribute all their energy to the "open" state only, extending the network lifetime as long as possible.

9.2.2 General Scenarios

The special case study in the previous section illustrates how the proposed sensor selection scheme with SUI prolongs the network lifetime by better utilizing the low cost sensors under specific QoS requirements. In this section, we extend the application QoS requirements to a general description, where sensors are randomly located in the warehouse, cooperating with each other to meet the applications coverage requirements. Consequently, the possible sensor sets vary with every random generation of the sensors locations. To examine the performance of the proposed sensor selection scheme, we discuss two different scenarios separately. One scenario considers homogeneous networks, where only one type of sensor is used. The other scenario considers heterogeneous networks, where multiple sensing variables are of interest to the application. Network lifetimes supported by each sensor selection scheme are compared to the optimal network lifetime, which is calculated by solving an optimization problem with the knowledge of each sensors initial lifetime and the exact a-posteriori probability of occurrence of each system state over the network lifetime. Practically, however, the exact probability of occurrence of each system state will vary over time and is hard to predict. Thus this method cannot be used in practice.

9.2.2.1 Homogeneous Networks

In this scenario, 10 motion sensors, as specified in Table 2, are randomly located in a 10m10m warehouse. For simplicity, they are assumed to have circular sensing areas with the specified radius. The warehouse monitoring system has two system states. System state 1 requires 100% coverage of the warehouse, while system state 2 requires no less than 80% coverage of the warehouse.

Fig. 9.9 illustrates the network lifetime supported by random selection, minimum power selection, $1/E$ selection and the proposed SUI selection compared with the optimal network lifetime, as system state 1s probability of occurrence increases from 0.1 to 0.9. This figure shows that the proposed sensor selection scheme always achieves the longest network lifetime among all the schemes, and it approaches the optimal network lifetime closely. $1/E$ selection also performs well, much better than random selection and minimum power selection. The proposed scheme and $1/E$ selection have close performance because when the sensors are homogeneous, the proposed cost function for sensor k is

$$SUI(k) = \frac{SRIL(k)}{SRL(k)} = \frac{SRIL(k) \cdot P}{E(k)} \propto \frac{SRIL(k)}{E(k)} \tag{9.14}$$

where P is the same power consumption for every sensor, and $E(k)$ is the remaining energy of sensor k. Hence, losing the diversity of a sensors power consumption, the proposed scheme can only take advantage of the sensors SRIL to prolong the network lifetime compared with $1/E$ selection. On the other hand, minimum power selection does not use any diversity of a sensors SRIL or remaining energy, so it performs even

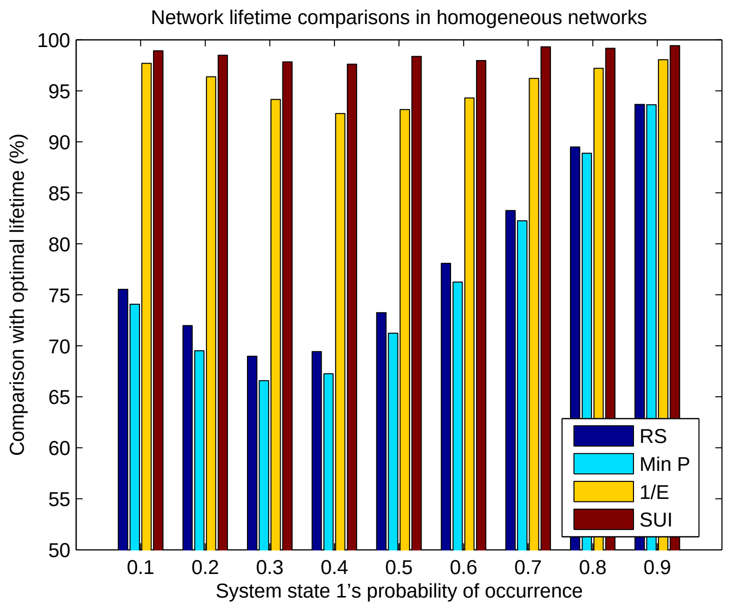


Figure 9.9: Network lifetime comparisons in homogeneous networks.

a bit worse than random selection. Another interesting phenomenon shown in Fig. 9.9 is that the 4 sensor selection schemes achieve their best performances simultaneously when system state 1s probability of occurrence is 0.9. At this point, most of the time the application works in system state 1, which requires 100% coverage of the warehouse and thus the redundancy in the network is low. Specifically, fewer possible sensor sets can support 100% coverage, and some sensors may be indispensable in all the possible sensor sets. Hence, the sensor selection scheme has fewer choices to improve the network lifetime, which is basically determined by these important sensors.

9.2.2.2 Heterogeneous Networks

In the heterogeneous scenario, 3 motion sensors, 3 vibration sensors and 3 sound sensors, as specified in Table 2, are randomly located in a 10 m $\times$ 10 m warehouse. Once again, they are assumed to have circular sensing areas with the specified radius. The applications QoS requirements remain the same as for the homogeneous network case, and again there are two system states: state 1 requires 100% coverage of the warehouse, while state 2 requires no less than 80% coverage of the warehouse.

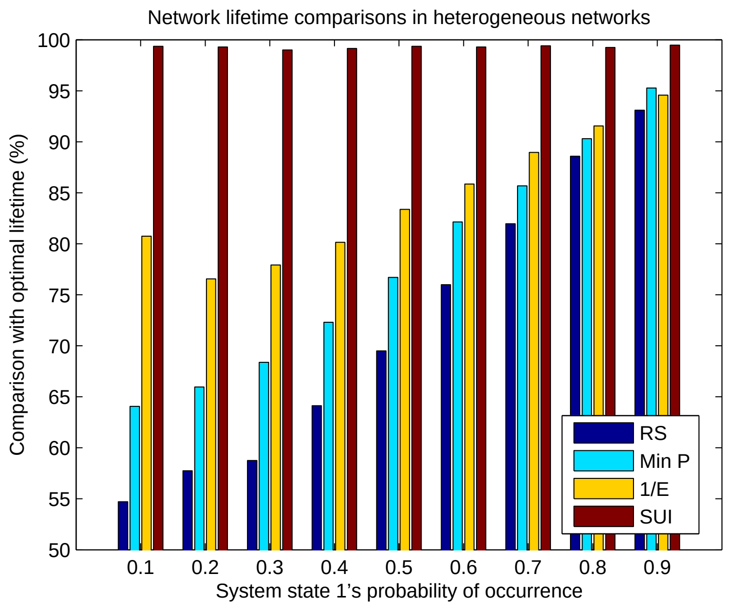


Figure 9.10: Network lifetime comparisons in heterogeneous networks.

Fig. 9.10 compares the performance of random selection, minimum power selection, $1/E$ selection and the proposed SUI selection scheme for heterogeneous networks. Even better performance is observed here than in the homogeneous scenario, with the proposed sensor selection scheme achieving 99% of the optimal network lifetime regardless of the system states probabilities of occurrence. The other three sensor selection schemes, however, provide much shorter network lifetime. Generally, in heterogeneous networks, sensors have different power consumption, remaining energy and also SRIL, which reflects the sensors contribution to the applications QoS. Neither minimum power selection nor $1/E$ selection incorporates even two of the three diversities. On the other hand, the proposed sensor selection scheme includes all of these factors into its cost function SUI, and hence always approaches the optimal network lifetime for both homogeneous and heterogeneous networks.

9.3 Summary

In this chapter, we propose a general sensor selection model to prolong network lifetime with QoS support for single-hop centralized networks. Our approach includes a 4-layer

QoS description model to abstract the application's QoS requirements and determine all possible sensor sets, and a sensor selection scheme that uses a cost function called SUI, which compares a sensor's relative ideal lifetime with its actual remaining lifetime. The proposed QoS description model simplifies the searching process for all possible sensor sets by decoupling an application's QoS requirements into simple logic functions.

Simulation results show that the proposed cost function and the corresponding sensor selection criteria consider different diversities in a sensor's power consumption, remaining energy and QoS capability, hence outperforming the existing sensor selection schemes in both homogeneous networks and heterogeneous networks. Furthermore, the network lifetime supported by the proposed sensor selection scheme closely approaches the optimal network lifetime. Compared to existing sensor selection schemes, our approach has clear advantages, especially for heterogeneous, single-hop networks, as it extends network lifetime with QoS support.

Chapter 10

Conclusions and Future Research

This dissertation investigates different sensor sleeping strategies in wireless sensor networks through comprehensive simulations as well as theoretical modeling and optimization. We are convinced by our theoretical and simulation results that the goal of dynamically controlling the sleeping strategy of wireless sensor networks through an adaptive sensor sleeping solution can be achieved, providing a significant improvement in network lifetime.

10.1 Summary of Contributions

The contributions of this dissertation are summarized as follows.

(1) I provide a performance study of various sensor sleeping strategies. Simulation results show that dynamically choosing an appropriate sensor sleeping strategy is promising in further prolonging the network lifetime.

(2) A sleeping scheme for Directed Diffusion is proposed to significantly improve the network lifetime by allowing the nodes that are not involved in the data transmission to sleep periodically.

(3) A general method to analyze the performance of duty-cycled MAC protocols is proposed. This method can obtain throughput, delay, and energy consumption for both synchronized and asynchronous duty-cycled MAC protocols. Moreover, it can be used to optimize the protocol parameters for a desirable performance.

(4) Performance analysis and optimization for S-MAC, a synchronized duty-cycled MAC protocol for wireless sensor networks, using the proposed Markov models and the performance analysis method, is described.

(5) Performance analysis and optimization for X-MAC, an asynchronous duty-cycled MAC protocol for wireless sensor networks, using the proposed Markov model and the performance analysis method, is described.

(6) I propose Sleeping Multipath Routing to trade off reliability for lifetime by carefully deciding how many routing paths to activate and putting the redundant routing paths to sleep.

(7) I provide an adaptive sensor sleeping solution for Sleeping Multipath Directed Diffusion and S-MAC, based on the S-MAC performance analysis using the Markov model. The adaptive sensor sleeping solution prolongs the network lifetime with reliability guarantee under varying reliability requirements and network conditions.

(8) A general 4-layer QoS description model is proposed to efficiently obtain all possible sensor sets that can support an application's QoS requirements.

(9) A new cost function SUI and a set of sensor selection criteria based on SUI is proposed to put redundant source nodes to sleep. SUI greatly prolongs the network lifetime, especially in heterogeneous networks.

This dissertation investigates various sensor sleeping strategies at the application layer, at the routing layer, at the MAC layer, and at multiple layers, through modeling and simulations. Therefore, our conclusions will not only guide the design and implementation of practical sensor networks, but they also provide solutions to improve the network performance by dynamically choosing the best sleeping strategy under different network and application scenarios.

10.2 Future Research Directions

Future research in this area includes: (1) combining the adaptive sensor sleeping solution at the routing and MAC layers with the source node selection algorithm at the application layer, so that the source node redundancy can be utilized to improve the network lifetime; (2) implementing our proposed adaptive sensor sleeping solution on existing sensor network hardware to improve the performance of the real-life wireless sensor networks; and (3) extending our proposed adaptive sensor sleeping solution to other routing and MAC protocols.

10.2.1 Adaptive Sensor Sleeping with Smart Source Node Selection

The proposed adaptive sensor sleeping solution coordinates routing layer sensor sleeping with MAC layer sensor sleeping. However, source node selection can also put sensors to sleep at the application layer, as we described in Chapter 9. Different choices of activated source nodes may result in different network lifetimes when the adaptive sensor sleeping solution is used. Hence, a connection between the source node selection and the consequent routing redundancy should be determined to optimize the network lifetime by combining application layer sensor sleeping with the routing and MAC layer sensor sleeping.

10.2.2 Implementation of Adaptive Sensor Sleeping Solution

Implementing the adaptive sensor sleeping solution on real sensor nodes, such as Motes, has the following challenges. (1) A sensor's transmission range is not fixed but varies according to the channel conditions, hence a node that is supposed to receive a packet may not receive the packet in a real-life network. Meanwhile, the capture effect will influence the network performance. (2) A Mote obtains its remaining energy by measuring the battery voltage, which is determined by the battery's discharge curve. However, every battery has a different discharge curve, and hence the remaining energy obtained by the Mote may not properly indicate its actual remaining lifetime. Therefore, experiments should be conducted to verify if the remaining lifetime provided by the Motes is accurate enough to support the Sleeping Multipath Directed Diffusion. Otherwise, an artificial remaining energy must be set in each Mote, with the remaining energy of each Mote decreasing after sending or receiving a packet. (3) The sink Mote may not be computationally capable of performing path selection and calculating the timer parameters, and hence a computer attached to the sink Mote may be required to do the tasks and pass the results to the sink Mote in real-time.

10.2.3 Extending Adaptive Sensor Sleeping Solution to Other Protocols

In this dissertation, we propose an adaptive sensor sleeping solution for Sleeping Multipath Directed Diffusion and S-MAC based on the Sleeping Multipath Routing (Chapter 7) and Markov model for duty-cycled MAC protocols (Chapter 4). This adaptive sensor sleeping solution can be extended to other protocols, since our proposed Sleeping Multipath Routing and Markov model for duty-cycled MAC protocols are general models and thus are not limited to Directed Diffusion and S-MAC. For example, our Markov model can be used to estimate the performance of X-MAC and other asynchronous duty-cycled MAC protocols, and hence we can extend the adaptive sensor sleeping solution to support asynchronous duty-cycled MAC protocols. Meanwhile, we can also extend the adaptive sensor sleeping solution to other multipath routing protocols using the same multipath selection algorithm.

Bibliography

[1] N. Bulusu, S. Jha, “Wireless Sensor Networks, A System Perspective,” Artech House, Norwood, MA, 2005, pp.1-2.

[2] M. Connolly, F. O’Reilly, “Sensor Networks and the Food Industry,” in the REAL-WSN ’05 Workshop on Real-World Wireless Sensor Networks, 2005.

[3] R. Jedermann, C. Behrens, C. Gorecki, D. Westphal, J. Congil, R. Laur, W. Benecke, and W. Lang, “Linking RFIDs and Sensors for Logistical Applications,” in the 12th International Confernece on Sensors, 2005, pp. 317-322.

[4] http://www.princeton.edu/ mrm/zebranet.html.

[5] K. Romer, F. Mattern, “The Design Space of Wireless Sensor Networks,” in IEEE Wireless Communications, vol. 11, issue 6, Dec. 2004, pp. 54-61.

[6] A. Baggio, “Wireless Sensor Netowrks in Precision Agricuclture,” in the REAL-WSN ’05 Workshop on Real-World Wireless Sensor Networks, 2005.

[7] K. Sha, W. Shi, O. Watkins, “Using Wireless Sensor Networks for Fire Rescue Applications: Requirements and Challenges,” in IEEE International Conference on Electro/Information Tehcnology, 2006, pp. 239-244.

[8] F. Michahelles, P. Matter, A. Schmidt, and B. Schiele, “Applying Wearbale Sensors to Avalanche Rescue: First Experiences with a Novel Avalanche Beacon,” in Computers & Graphics, vol. 27, issue 6, Dec. 2003, pp. 839-847.

[9] A. Ko, H. Lau, “Robot Assisted Emergency Search ans Rescue System with a Wireless Sensor Networks,” in International Journal of Advanced Science and Technology, vol. 3, Feb. 2009, pp. 69-78.

[10] V. Jone, V. Gay, P. Leijdekkers, “Body Sensor Networks for Mobile Health Monitoring: Experience in Europe and Australia,” in the 4th International Conference on Digital Society, 2010, pp. 204-209.

[11] C. Chiasserini, R. Gaeta, M. Garetto, M. Gribaudo, D. Manini, M. Sereno, “Fluid Models for Large-scale Wireless Sensor Networks,” in Performance Evaluation, vol. 64, issue 7-8, 2007, pp. 715-735.

[12] J. Kulik, W. Rabiner, H. Balakrishnan, “Adaptive Protocols for Information Dissemination in Wireless Sensor Networks,” in 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking, 1999, pp. 174-185.

[13] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, “Directed Diffusion for Wireless Sensor Networking,” in IEEE/ACM Transactions on Networking, vol. 11, issue 1, 2003, pp. 2-16.

[14] J. Chang, L. Tassiulas, “Maximum Lifetime Routing in Wireless Sensor Networks,” in IEEE/ACM Transactions on Networking, vol.12, Issue 4, 2004, pp. 609-619.

[15] M. Perillo, W. Heinzelman, “DAPR: A Protocol for Wireless Sensor Networks Utilizing an Application-based Routing Cost,” in Wireless Communications and Networking Conference, vol. 3, 2004, pp. 1540-1545.

[16] H. Luo, J. Luo, Y. Liu, S. Das, “Adaptive Data Fusion for Energy Efficient Routing in Wireless Sensor Networks,” in IEEE Transactions on Computers, vol. 55, issue 10, 2006, pp. 1286-1299.

[17] M. Zoghi, M. Kahaei, “Sensor Selection for Target Tracking in WSN Using Modified INS Algorithm,” in 3rd Internatioanl conference on Information and Communication Technologies: From Theory to Applications, 2008, pp. 1-6.

[18] Y. Xu, J. Heidemann, D. Estrin, “Geography-informed Energy Conservation for Ad Hoc Routing,” in the 7th Annual International Conference on Mobile Computing and Networking, 2001, pp. 70-84.

[19] B. Chen, K. Jamieson, H. Balakrishnan, R. Morris, "Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wirleess Networks," in Wireless Networks, voil. 8, issue 5, Kluwer Academic Publishers, Hingham, MA, 2002, pp. 481-494.

[20] R. Zheng, R. Kravets, "On-demand Power Management for Ad Hoc Networks," in the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies, vol.1, 2003, pp. 481-491.

[21] H. Wang, W. Wang, D. Peng, H. Sharif, "A Route-oriented Sleep Approach in Wireless Sensor Network," in the 10th IEEE Singapore International Conference on Communication Systems, 2006, pp. 1-5.

[22] W. Ye, J. Heidemann, D. Estrin, "Medium Access Control with Coordinated Adaptive Sleeping for Wireless Sensor Networks," in IEEE/ACM Transactions on Networking, vol. 12, issue 3, 2004, pp. 493-506.

[23] J. Polastre, J. Hill, D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," in the 2nd International Conference on Embedded Networked Sensor Systems, 2004, pp. 95-107.

[24] H. Cheng, X. Jia, "An Energy Efficient Routing Algorithm for Wireless Sensor Networks," in International Conference on Wireless Communications, Networking and Mobile Computing, vol. 2, 2005, pp. 905-910.

[25] http://circuit.ucsd.edu/ curts/courses/ECE284_F05/lectures /ECE284_F05_L07_EnergyEfficiency_2pp.pdf

[26] O. Yang, and W. Heinzelman, "A Better Choice for Sensor Sleeping," in the 6th European Conference on Wireless Sensor Networks, 2009, pp. 134-149.

[27] O. Yang, and W. Heinzelman, "Modeling and Performance Analysis for Duty-cycled MAC Protocols in Wireless Sensor Networks," accepted by IEEE Transactions on Mobile Computing.

[28] O. Yang, and W. Heinzelman, "Modeling and Throughput Analysis for SMAC with a Finite Queue Capacity," in International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP) 2009.

[29] O. Yang, and W. Heinzelman, “Modeling and Throughput Analysis for X-MAC with a Finite Queue Capacity,” in Global Communication Conference (GlobeCom) 2010.

[30] O. Yang, and W. Heinzelman, “Sleeping Multipath Routing: A Trade-off Between Reliability and Lifetime in Wireless Sensor Networks,” submitted to GlobeCom, 2011.

[31] O. Yang, and W. Heinzelman, “Sensor Selection Cost Function to Increase Network Lifetime with QoS Support,” in ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM) 2008.

[32] G. Xing, X. Wang, Y. Zhang, C. Lu, R. Pless, C. Gill, “Integratd Coverage and Connectivity Configuration for Energy Conservation in Wireless Sensor Networks,” in ACM Transactions on Sensor Networks, vol. 1, issue 1, 2005, pp. 36-72.

[33] T. Dam, L. Langendoen, “An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Netowrks,” in the 1st International Conference on Embedded Networked Sensor Systems, 2003, pp. 171-180.

[34] M. Buettner, G. Yee, E. Anderson, and R. Han, “X-MAC: a Short Preamble MAC Protocol for Duty-cycled Wireless Sensor Networks,” in SenSys, 2006, pp. 307320.

[35] K. Wong, D. Arvind, “SpeckMAC: Low-power Decentralised MAC Protocol Low Data Rate Transmissions in Specknets,” in REAL-MAN06, 2006.

[36] H. Wang, X. Zhang, F. Nat-Abdesselam, and A. Khokhar, ‘DPS-MAC: An Asynchronous MAC Protocol for Wireless Sensor Networks,” in HiPC07, 2007, pp. 393-404.

[37] J. Zhang, F. Nat-Abdesselam, B. Bensaou, “Performance Analysis of an Energy Efficient MAC Protocol for Sensor Networks,” in the International Symposium on Parallel Architectures, Algorithms, and Networks, 2008, pp. 254-259.

[38] S. Hong and H. Kim, “A Multi-hop Reservation Method for End-to-End Latency Performance Improvement in Asynchronous MAC-based Wireless Sensor Networks,” in IEEE Trans. Consumer Electronics, vol. 55, no. 3, Aug. 2009, pp. 1214-1220.

[39] K. Kumar, P. Kumar, "Tmote implementation of BMAC and SMAC protocols," http://www.cse.iitk.ac.in/users/vkirankr/wireless_report.pdf

[40] V. Tippanagoudar, I. Mahgoub, A. Badi, "Implementation of the Sensor-MAC protocol for the JiST/SWANS simulator," in IEEE/ACS International Conference on Computer Systems and Applications, 2007, pp. 225-232.

[41] G. Bianchi, "IEEE 802.11 saturation throughput analysis." in IEEE Communication Letters, vol. 1, no. 12, Dec. 1998, pp. 318-320.

[42] C. Fischione, S. Coleri Ergen, P. Park, K. H. Johansson, A. Sangiovanni-Vincentelli, "Medium Access Control Analytical Modeling and Optimization in Unslotted IEEE 802.15.4 Wireless Sensor Networks," in SECON 2009, pp. 1-9.

[43] Y. Zhang, C. He, and L. Jiang, "Energy and QoS Trade-off Analysis of S-MAC protocol in Wireless Sensor Networks," in CCWMSN, 2007, pp. 76-79.

[44] Y. Wang, M. Vuran, S. Goddard, "Cross-layer Analysis of the End-to-end Delay Distribution in Wireless Sensor Networks," in the 30th IEEE Real-Time Systems Symposium, 2009, pp. 138-147.

[45] C. Wang, Y. Sun, and H. Ma "Analysis of Data Delivery Delay in Acoustic Sensor Networks," in 2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, pp. 283-287.

[46] J. Rousselot, A. El-Hoiydi, and J. Decotignie, "Low Power Medium Access Control Protocols for Wireless Sensor Networks," in the 14th European Wireless Conference, 2008, pp.1-5.

[47] J. Luo, L. Jiang, C. He, "Finite Queuing Model Analysis for Energy and QoS Tradeoff in Contention-Based Wireless Sensor Networks," in IEEE International Conference on Communications, 2007, pp. 3901-3906.

[48] J. Luo, L. Jiang, C. He, "Performance Analysis of Synchronous Wakeup Patterns in Contention-based Sensor Networks Using a Finite Queuing Model," in IEEE GLOBECOM, 2007, pp. 1334-1338.

[49] H. Wu, Y. Peng, K. Long, S. Cheng, and J. Ma, "Performance of reliable transport protocol over IEEE 802.11 wireless LAN: analysis and enhancement," in INFOCOM, 2002, pp. 599-607.

[50] Y. Fallah, F. Agharebparast, M. Minhas, H. Alnuweiri, and V. Leung, "Analytical modeling of contention-based bandwidth request mechanism in IEEE 802.16 wireless networks," in IEEE Trans. Vehicular Technology, vol. 57, no.5, Sep. 2008, pp. 3094-3106.

[51] J. He, Z. Tang, H. Chen, and Q. Zhang, "An accurate and scalable analytical model for IEEE 802.15.4 slotted CSMA/CA networks, in IEEE Trans. Wireless Communications, vol. 8, no. 1, Jan. 2009, pp. 440-448.

[52] A. Alshanyour, A. Agarwal, "Three-Dimensional Markov Chian Model for Performance Analysis of the IEEE 802.11 Distributed Coordination function," in IEEE GLOBECOM 2009, pp. 1-7.

[53] G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Cordination Function," in IEEE JSAC, vol. 18. no. 3, March 2000, pp. 535-547.

[54] H. Chhaya, S. Gupta, "Perofrmance Modeling of Asynchronous Data Transfer methods of IEEE 802.11 MAC Protocol," in Wireless Networks, vol. 3, issue 3, Kluwer Academic Publisher, 1997, pp. 217-234.

[55] Y. Liaw, A. Dadej, A. Jayasuriya, "Performance Analysis of IEEE 802.11 DCF under Limited Load," in Asia-Pacific Conference on Communications, 2005, pp. 759-763.

[56] S. Dulman, T. Nieberg, J. Wu, and P. Havinga, "Trade-Off between Traffic Overhead and Reliability in Multipath Routing for Wireless Sensor Networks," in Wireless Communications and Networking, 2003, pp. 1918-1922.

[57] B. Yahya and J. Ben-Othman, "REER: Robust and Energy Efficient Multipath Routing Protocol for Wireless Sensor Networksm," in Global Telecommunications Conference, 2009, pp. 1-7.

[58] L. He, “Efficient Multi-Path Routing in Wireless Sensor Networksm,” in the 6th International Conference on Wireless Communications Networking and Mobile Computing, 2010, pp. 1-4.

[59] A. Chamam, S. Pierre, “Dynamic sensor activation for maximizing network lifetime under coverage constraint,” in International Symposium on Communications and Information Technologies, 2007, pp. 971-976.

[60] J. Chang, L. Tassiulas, “Energy conserving routing in wireless ad hoc networks,” in the 19th Annual Joint Conference of IEEE Computers and Communications Societies, 2000, pp. 22-31.

[61] C. Chen, S. Tekinay, C. Saraydar, “Minimum-power & energy-balancing cellular ad hoc augmented networks,” in IEEE Wireless Communications and Networking Conference, 2004, pp. 1099-1103.

[62] S. Das, W. Choi, “Coverage-adaptive random selection with latency control for application-specific data gathering in wireless sensor networks,” inInternational Conference on Control and Automation, 2005, pp. 214-219.

[63] B. Lo, S. Thiemjarus, R. King, G. Yang, “Body Sensor Network - a Wireless Sensor Platform for Pervasive Healthcare Monitoring,” Demonstrations in the 3rd International Conference on Pervasive Computing, 2005.

[64] R. Mushini, D. Simon, “On Optimization of Sensor Selection for Aircraft Gas Turbine Engines,” in the 18th International Conference on Systems Engineering, 2005, pp. 9-14.

[65] U. Ramdaras, F. Absil, “Networks of Maritime Radar Systems: Sensor Selection Algorithm for PD¡1 Based on the Modified Riccati Equation,” in IEEE Non-linear Statistical Signal Processing Workshop, 2006.

[66] P. Sharma, A. Narasimhan, S. Ramalingam, and S. Tripathi, “Energy Conservation in Sensor Networks Through Selective Node Activation,” in International Symposium on a World of Wireless, Mobile and multimedia Networks, 2006.

[67] S. Singh, M. Wu, S. Raghavendra, “Power Aware Routing in Mobile Ad Hoc Networks,” in the 4th Annual International Conference on Mobile Computing and Networking, 1998, pp. 181-190.

[68] C. Tessier, M. Berducat, R. Chapuis, and F. Chausse, “A New Landmark and Sensor Selection Method for Vehicle Localization and Guidance,” in Intelligent Vehicles Symposium, 2007, pp. 123-129.

[69] Y. Zou, K. Chakrabarty, “A Distributed Coverage- and Connectivity- Centric Technique for Selecting Active Nodes in Wireless Sensor Networks,” in IEEE Trans. on Computers, vol. 54, no. 8, Aug. 2005, pp. 978-991.

[70] IEEE 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. (2007 revision). IEEE-SA. 12 June 2007.

[71] http://www.isi.edu/nsnam/ns/

[72] http://etd.adm.unipi.it/theses/
available/etd-05252004-154652/unrestricted/Chap4.pdf

[73] http://www.xbow.com/Products/Product_pdf_files
/Wireless_pdf/MICAz_Datasheet.pdf

[74] C. Merlin, “Adaptability in Wireless Sensor Networks Through Cross- Layer Protocols and Architectures,” Ph.D. Dissertation, Dept. ECE, University of Rochester, Rochester, NY, 2009.

[75] M. Demirbas, K. Chow, and C. Wan, “INSIGHT: Internet-sensor Integration for Habitat Monitoring,” in International Symposium on a World of Wireless, Mobile and Multimedia Networks, 2006.

[76] T. Hsieh, “Using Sensor Networks for Highway and Traffic Applications,” in IEEE Potentials, vol. 23, no. 2, April-May 2004, pp. 13-16.

[77] http://www.digikey.com/scripts/dksearch
/dksus.dll?Pname?Name=359−1001−ND&site=us&dkcid= 10.

[78] http://www.isthq.com.

[79] http://www.ovt.com/data/parts/pdf/OV6680_PB(1.01)_web.pdf.

[80] J. Kumagai, "Life of Birds [Wireless Sensor Network for Bird Study]," in IEEE Spectrum, vol. 41, no. 4, April 2004, pp. 42-49.

[81] J. Lee, B. Krishnamachari, and C. Kuo, "Impact of Heterogeneous Deployment on Lifetime Sensing Coverage in Sensor Networks," in the 1st IEEE Annual Communications Society Conference on Sensor and Ad Hoc communications and Networks, 2004, pp. 367-376.

[82] K. Liska, "A Sensor Network Architecture for Aardiac Health Monitoring," in the 4th IEEE Consumer Communications and Networking Conference, 2007, pp. 737-740.

[83] S. Ahmad, R. Eskicioglu, P. Graham, "Design and Implementation of a Sensor Network Based Location Determination Service for Use in Home Networks," in IEEE International Conference on Mobile Adhoc and Sensor Systems, 2006, pp. 622-626.

[84] E. Bjornemo, M. Johansson, and A. Ahlen, "Two Hops Is One Too Many in an Energy-limited Wireless Sensor Network," in IEEE International Conference on Acoustics, Speech and Signal Processing, 2007, pp. 181-184.

[85] S. Brown, and Z. Vranesic, "Fundamentals of Digital Logic with VHDL Design," TATA McGraw-Hill, New Delhi, 2003, pp. 144-165.

[86] K. Schwieger, and G. Fettweis, "Multi-hop Transmission: Benefits and Deficits," in the GI/ITG Fachgesprch "Sensornetze", 2004.

[87] Single-chip 2.4 GHz IEEE 802.15.4 Compliant and ZigBee(TM) Ready RF Transceiver - CC2420. http://focus.ti.com/docs/prod/folders/print/cc2420.html.

[88] A. Tiwari, F. Lewis, and S. Ge, "Wireless Sensor Networks for Machine Condition Based Maintenance," in the 8th Control, Automation, Robotics and Vision Conference, 2004, pp. 461-467.

[89] L. Zhong, J. Rabaey, and A. Wolisz, “Does Proper Coding Make Single Hop Wireless Sensor Networks Reality: The Power Consumption Perspective,” in IEEE Wireless Communications and Networking Conference, 2005, pp. 664-669.