UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

ENERGY-EFFICIENT PROTOCOL DESIGN AND ANALYSIS FOR WIRELESS SENSOR NETWORKS

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By

TAO ZHENG
Norman, Oklahoma
2011

ENERGY-EFFICIENT PROTOCOL DESIGN AND ANALYSIS
FOR WIRELESS SENSOR NETWORKS

A DISSERTATION APPROVED FOR THE
SCHOOL OF COMPUTER SCIENCE

BY

Dr. Sridhar Radhakrishnan, Chair

Dr. John K. Antonio

Dr. S. Lakshmivarahan

Dr. Sudarshan Dhall

Dr. Suleyman Karabuk

Acknowledgements

I would like to express my deepest gratitude to my advisor, Dr. Sridhar Radhakrishnan, for his expert guidance and inspiration without which this work would not have been possible. Not only he taught me research skills, but he always encouraged me to think independently. I am also deeply grateful for his patience and friendship throughout my graduate studies.

I would also like to thank Dr. Venkatesh Sarangan, a former professor at the Oklahoma State University, for his insightful comments and constructive criticisms on my research work. His comments always helped to enrich my ideas and to improve my work. He also helped me on improving the writings of a few co-authored papers.

I owe special thanks to Dr. S. Lakshmivarahan, for whom I was a teaching and research assistant for more than two years. I deeply appreciate his support, inspiration and friendship throughout my graduate studies.

My thanks also go out to the other members of my dissertation committee, Dr. John K. Antonio, Dr. Sudarshan K. Dhall and Dr. Suleyman Karabuk, for their understanding, patience and input.

I wish to thank all the professors with whom I had taken classes. Those classes had laid solid foundation for my research work. Thank Barbara Bledsoe, Chyrl Yerdon, Jim Summers and all other staff members for their kind help. I also thank the former members of my research group, especially Shankar M. Banik, Aravind B. Mohanoor, Jonyung Kim and Waleed Numay, for exchanging ideas with me on

different research topics. Thank all of my friends in Norman for their continuous support and encouragement.

I also thank my parents and sisters for their understanding of my overseas academic pursuit and their unwavering faith in my abilities. Last but not least a big thanks goes to my wife, Yan Li. She always stood by me through my good times and bad. I would never have been able to finish my dissertation without her love, care and sacrifice.

Table of Contents

List of Tables

List of Figures

Abstract

Wireless sensor networks are an emerging technology which has the promise of revolutionizing the way of collecting, processing and disseminating information. Due to the small sizes of sensor nodes, resources like battery capacity, memory and processing power are very limited. Wireless sensor networks are usually unattended once deployed and it is infeasible to replace batteries. Designing energy-efficient protocols to prolong the network life without compromising too much on the network performance is one of the major challenges being faced by researchers.

Data generation in wireless sensor networks could be bursty as it is dictated by the presence or absence of events of interest that generate these data. Therefore sensor nodes stay idle for most of the time. However, idle listening consumes as much energy as receiving. To save the unnecessary energy consumption due to idle listening, sensor nodes are usually put into sleep.

MAC protocols coordinate data communications among neighboring nodes. We designed an energy-efficient MAC protocol called PMAC in which sleep-awake schedules are determined through pattern exchange. PMAC also adapts to different traffic conditions.

To handle bursty traffic and meanwhile preserve energy, dual radio interfaces with different ranges, capacity and power consumption can be employed on each individual sensor node. We designed a distributed routing-layer switch agent which intelligently directs traffic between the dual radios. The low-power radio will be

used for light traffic load to preserve energy. The high-power radio is turned on only when the traffic load becomes heavy or the end-to-end delay exceeds a certain threshold. Each radio has its own routing agent so that a better path can be found when the high-power radio is in use.

Data gathering is a typical operation in wireless sensor networks where data flow through a data gathering tree towards a sink node. DMAC is a popular energy-efficient MAC protocol specifically designed for data gathering in wireless sensor networks. It employs staggered sleep-awake schedules to enable continuous data forwarding along a data gathering tree, resulting in reduced end-to-end delays and energy consumption. we have analyzed end-to-end delay and energy consumption with respect to the source node for both constant bit rate traffic and stochastic traffic following a Poisson process. The stochastic traffic scenario is modeled as a discrete time Markov chain and expressions for state transition probabilities, the average delay and average energy consumption are developed and are evaluated numerically. Simulations are carried out with various parameters and the results are in line with the analytical results.

Lots of work had been done on constructing energy-efficient data gathering trees at the routing layer. We proposed a sleep scheme at the routing layer called DGSS which could be incorporated into different data gathering tree formation algorithms. Unlike DMAC, in which nodes are scanned level by level, DGSS starts scanning from the leaf nodes and shrinks inward towards the sink node. Simulation shows

that DGSS can achieve better energy efficiency than DMAC at relatively higher data rates.

Chapter 1

Introduction

1.1 Wireless Sensor Networks

Wireless Sensor Networks (WSNs) are application-specific wireless *ad hoc* networks consisting of nodes equipped with sensors, computing devices and wireless communication devices. In recent years, the rapid advances in low-cost, low-power circuit design have enabled the development of WSNs and made it one of the emerging technologies that may change the world one day. WSNs are envisioned as a paradigm shift [24] from the traditional information collection and have the promise of revolutionizing the way of collecting, processing and disseminating information about environment. The distributed nature of WSNs makes it more fault-tolerant. The *ad hoc* nature of WSNs allows a fast deployment of the network, making it attractive to military applications and hostile environment.

Many different types of sensors, such as light, thermal, acoustic, magnetic, mechanical sensors, can be equipped on sensor nodes. That empowers WSNs to have a wide range of potential applications in various fields such as environmental monitoring [11], habitat monitoring [34, 61], structural monitoring [65, 28], precision

agriculture [31], weather condition, health care [51], battlefield surveillance [7] and homeland security, etc.

1.2 Unique Features and Challenges

While WSNs share many commonalities with traditional *ad hoc* networks, they also have a few unique features which open specific challenges to researchers. Those unique features and challenges include [25]:

- **Application specific** WSNs are not generic purpose networks like the Internet, but application-specific. Different applications might require different sensing and communication technologies. For instance, some applications might be delay-sensitive and some might not at all; some applications may tolerate data loss and some others may not. It is unlikely to have generic solutions for all the different applications with very different requirements. Since WSNs are heavily driven by the application, the protocols developed need to be tailored, to a large extent, for the application that a particular network is built to address.

- **Self organization** The *ad hoc* nature of WSNs requires sensor nodes in a WSN to coordinate with each other and organize themselves (with little or no external help) into a fully functional network, which can then relay the sensed information back to the data collecting sink nodes.

- **Scarce resources** The size of sensor nodes limits the resources available to sensor nodes, such as battery power, memory, computational power. The limitations of memory and computational power might be mitigated soon with the rapid development of fabrication techniques. The energy constraint, however, is unlikely to be solved in the near future with the slow progress in improving battery capacity. Sensor nodes are usually left unattended after deployment and it is infeasible to either replace or recharge their batteries due to the large scale of WSNs. Hence, energy-efficient protocols are needed for prolonging network lifetime. Due to the scarce resources, the protocols designed for WSNs should also be kept as simple as possible.

- **Large scale** WSNs may consist of thousands or even millions of tiny sensor nodes. The large scale of WSNs requires the protocols for WSNs must be more scalable, compared to the current *ad hoc* networks. Hierarchical architecture, localized algorithms, and data aggregation might have to be used to achieve high scalability.

- **Bursty traffic** Unlike the traditional network mostly driven by human, WSNs are driven by environmental events in most of applications. Therefore, the data flow is very bursty — low data rates during most of the time and high data rates when events occur.

- **Data centric** This represents a paradigm shift [22] from traditional networks. Traditional networks are *address-centric*, in which we know where to get the

data we are interested in. While in WSNs we only know things we are interested in, but do not know where they are located. In other words, WSNs tend to operate as a collective structure, rather than supporting many independent point-to-point flows [63].

- **Data redundancy** This requires in-network processing such as data aggregation or data fusion to reduce the amount of data flowing around in the networks.

1.3 Energy Savings in Wireless Sensor Networks

Since energy constraint is one of the most challenging issues WSNs are facing, energy saving surely becomes one of the primary research topics on WSNs. Numerous energy saving techniques and protocols have been proposed at different layers of the OSI (Open Systems Interconnection) model. We now review some of the work at each layer along the OSI protocol stack to understand those techniques and challenges related to energy savings.

1.3.1 Physical Layer

Physical layer includes the physical devices, communication channels and topology control. Low-power hardware design, modulation and encoding schemes are critical for saving energy in sensing, data processing and communication. Since topology

control is more relevant to protocol design, we will focus on the energy efficient formation of network topology.

Topology control or power control usually refers to the construction of network topology by adjusting the transmit power and hence the transmission range of each node. In other words, the *per node transmit power* is determined as the result of topology control [47]. Since the formation of the initial topology has direct impact on the degree (number of neighbors) of a sensor node, which in turns has impact on the interferences and data redundancy, improper topology will have negative impact on the energy consumption. Therefore, energy efficient topology control schemes are vital in maximizing the network lifetime. The goal in topology control is to have each node transmit at the lowest possible power while preserving network connectivity and the robustness to node failures. Transmitting at unnecessarily high power not only reduces the lifetime of the nodes and the network, but also introduces excessive interference, which reduces the network capacity and increases the end-to-end delay.

Topology control has been well studied in wireless *ad hoc* networks [16, 47, 26] and has been formulated as network optimization problems. The formulation methods vary with different optimization metrics and constraints. Both [16] and [15] target at minimizing the *total transmit power* and meanwhile maintaining strong network connectivity. However, [16] assumes there exists unidirectional links in the network, while [15] assumes only bidirectional links exist in the network. Authors in both papers proved their problems are NP-Complete and hence heuristics were used. One

of the heuristic methods is to assign power based on minimum spanning tree, which gives an approximation ratio of 2.

In [47], topology control is formulated as a constrained optimization problem whose objective is to minimize the *maximum transmit power* with the constraints to maintain connectivity or bi-connectivity. Optimal solutions were presented for both connected and bi-connected static networks. Distributed algorithms based on simple heuristics were presented for mobile networks. Consequently, there is no guarantee on network connectivity.

The drawback in the above formulations is that they all assume the networks are static and the coordinates or the distances among all the nodes are known before hand and are fixed, which may not be realistic in WSNs. Localized and distributed algorithms [48, 62] are better candidates for WSNs. [62] proposed a distributed cone-based algorithm which does not need know the global position information of the nodes in the network but only local information instead. Each node makes local decisions on its transmit power in such a way that they collectively guarantee global connectivity. The algorithm assumes that all nodes know each other's direction through message exchange. Two phases are involved in the algorithm. The first phase is to find a connected graph by letting each node continue grow its transmit power until it finds at least one neighbor for any cone with angle α or it hits the maximum transmit power. The second phase is to reduce the node degrees, in which redundant edges are removed. This helps in reducing node interferences and improving throughput.

1.3.2 MAC Layer

In WSNs, sensor nodes communicate with their neighbors through shared wireless channels. If multiple nodes in a neighborhood send data through the shared channel at the same time, collision occurs and data will be garbled. Therefore, some kind of coordination mechanisms need be in place for such one-hop communications. MAC (Medium Access Control) layer protocols provide such mechanisms by deciding which node(s) should transmit first. MAC layer is a sub-layer of the Logical Link Control (LLC) layer in the OSI model, as only one-hop communication is concerned. MAC layer protocols can be divided into contention-free and contention-based MAC protocols.

As mentioned earlier, bursty traffic is one of the unique features of WSNs. Radios on sensor nodes stay in the idle-listening state for most of the time since heavy traffic loads only present when events occur. It is a known fact that most of radio devices consume as much power in the idle-listening state as in the receiving state. Significant amount of energy will be wasted if radios are left in such idle-listening state. Many energy efficient MAC protocols have been proposed for WSNs, trying to bring down the *per node idle-listening power consumption* by lowering the duty-cycles of radios through sleep scheduling. In other words, radios are put into sleep if they are not involved in any data communication.

SMAC [71] is a well-known energy efficient MAC protocol specifically designed for WSNs. It forces sensor nodes operate at low duty cycle by putting them into

sleep periodically instead of idle listening. Sensor nodes also sleep during overhearing to save power. Although SMAC saves power, it does not adapt to network traffic well since it uses a fixed duty cycle for all the sensor nodes. A duty cycle tuned for heavy traffic loads results in energy wastage when the traffic is light, while duty cycle tuned for light traffic loads results in low throughput under heavy traffic loads. SMAC with adaptive listening [72] was proposed later on to achieve adaptive duty cycles. The Timeout-MAC protocol (TMAC) [56] improves SMAC in [71] by introducing adaptive duty cycles. If there is no activity in the vicinity of a node for a time T_A, the node goes to sleep. Such an adaptation frees the application from the burden of selecting an appropriate duty cycle. TMAC has the same performance as SMAC under constant traffic loads, but saves more energy under variable traffic. The downside of TMAC's aggressive power conserving policy is that nodes can go to sleep rather early, resulting in increased latency and lower throughput. Another drawback in both SMAC and TMAC is that, they group the communication during small periods of activity. As a result, the protocols collapse under heavy traffic loads. Data-gathering MAC (DMAC) [32] is another protocol that uses adaptive duty cycles. It yields low end-to-end delay in convergecast communication by staggering the sleep-awake schedules of the nodes at different levels of the data gathering tree. DMAC outperforms SMAC in terms of latency, throughput and energy efficiency for low-rate data gathering.

Above are the examples of coordinated or synchronous duty cycling, in which sleep-awake is synchronized across all nodes, or subsets thereof. Distributed time

synchronization [17] is usually needed in this case and it is quite challenging to achieve for large scale networks like WSNs. As a contrast, uncoordinated or asynchronous duty cycling establishes sleep schedules without any explicit coordination. B-MAC [45] and X-MAC [8] are examples of such asynchronous duty-cycling protocols. In B-MAC, sensor nodes wake up periodically to check if there is any activity currently on the channel. If so, the sensor nodes stay awake to receive any possible incoming traffic. Whenever a sensor node wants to send data, a long preamble is sent out first. The preamble lasts longer than the receiver's sleep interval to ensure that the receiver is awake upon any data transmission. B-MAC can achieve very low duty cycle under light traffic, as only a short period of time is needed in sensing the channel activity within every awake time. However, B-MAC could stay awake unnecessarily due to overhearing traffic bound for other nodes. X-MAC tried to solve this overhearing problem by using a sequence of short preambles instead of a really long one. In contrast to B-MAC and X-MAC, where the sender initiates a preamble to ensure the receiver is ready to receive, RI-MAC [53] uses a receiver initiated scheme, where the sender waits for a beacon or signal from the receiver to start data transmission.

1.3.3 Routing Layer

Routing protocols define the paths to relay data packets. As for wireless *ad hoc* networks, routing protocols for WSNs can be classified as proactive (or table-driven)

protocols, reactive (or on-demand) protocols and hybrid (both proactive and reactive) protocols. Proactive routing protocols try to maintain an up-to-date map of the network, by continuously evaluating known routes and attempting to find out new ones. This task is realized by sending reliable and up-to-date routing information from each sensor node to every other node in the network. Unlike proactive routing protocols, reactive protocols only start a route discovery procedure when needed. In this case, a sort of global search procedure is started. Although it does not require constant updates to be sent throughout the network, as in pro-active protocols, this process does cause delays, since the requested routes are not available and have to be found. For WSNs, we need factor in the unique features of WSNs, such as limited resources and large scale. This results in the existence of numerous routing protocols specifically for WSNs. A new routing paradigm called *data-centric* routing has been introduced in the directed diffusion paper [22]. In traditional wired networks, people do know where the information they want is. Therein the routing is to find a path between a pair of addressable end nodes. This is so called *address-centric* routing. While in a WSN, the data people are interested in may scatter all around the network and people have no idea where those data are located. The *data-centric* routing is to find the data of interest and consolidate them if necessary.

As far as energy consumption concerns, one of the goals is to design energy efficient routing protocols which minimize the *per route/flow power consumption*. However, finding the minimum energy path and using it for every communication are not the best thing to do for prolonging network lifetime, as the frequent usage of

a low energy path leads to energy depletion at the nodes along that path and in the worst case may lead to network partition. To prevent the optimal path from getting depleted, it is necessary to use sub-optimal paths sometimes in order for the network to degrade gracefully as a whole rather than getting partitioned. In other words, the energy dissipation is more evenly distributed or balanced among the sensor nodes in the network. In [49], multiple paths are found between source and destination, and each path is assigned a probability of being chosen, depending on the energy metric. Every time data are to be sent from the source to destination, one of the paths is randomly chosen depending on the probabilities. This ensures that none of the paths is used all the time, preventing from energy depletion.

[41] proved that the routing problem in WSNs so as to maximize network lifetime is NP-hard, where the lifetime is defined as the number of successful routing requests before the first failed routing request. A two-step heuristic algorithm is developed to maximize the network lifetime. The basic idea is to delay the energy depletion at a node as much as possible. [12] formulates the routing problem as a linear programming problem, where the objective is to maximize the network lifetime defined as the time when the network partition happens due to battery outage. A shortest path routing algorithm is proposed based on link costs reflecting both the communication energy consumption rates and the residual energy levels at the two end nodes. Simulation shows that the routing algorithm can achieve network lifetime that is very close to the optimal network lifetime obtained by solving the linear programming problem. [66] proposed a novel utility-based nonlinear optimization

formulation to the maximum lifetime routing problem. Based on this formulation, a fully distributed and localized routing algorithm was presented and was proved to converge to the optimal point where the network lifetime is maximized.

Kim and Liu [27] studied the routing problem in WSNs where sensors are duty-cycled, which is quite common in WSNs. When sensors alternate between on and off modes, delay encountered in packet delivery due to loss in connectivity can become a serious problem, and how to achieve delay-optimality is non-trivial. For instance, when sensor nodes' sleep-awake schedules are uncoordinated, it is not immediately clear whether a sensor node with data to transmit should wait for a particular neighbor (who may be on a short route) to become active before transmission, or simply transmit to an active neighbor to avoid waiting. To obtain some insight into this problem, the authors formulate the above problem as an optimal stochastic routing problem, where the randomness in the system comes from random duty cycling, as well as the uncertainty in packet transmission due to channel variations. Some existing optimal routing algorithms are no longer optimal when duty cycling is introduced. An optimal centralized stochastic routing algorithm was developed for randomly duty-cycled WSNs, and a distributed algorithm utilizing local sleep-awake schedules was also presented in the paper.

A good survey on the routing techniques for WSNs is available in [5].

1.3.4 Transport Layer

Transport layer usually provides the end-to-end reliability as it sits on top of an unreliable network layer. Congestion control is another duty of the transport layer. Congestion control significantly improves the end-to-end throughput and delay by throttling down the date rates at source nodes when congestion is detected. In WSNs, data gather from multiple sensor nodes to a single or few sink nodes. This unique traffic pattern in WSNs puts heavy burden on the relaying sensor nodes near the sink(s). Congestion at those nodes will cause long end-to-end delay, low throughput and unnecessary energy wastage. Therefore, congestion or rate control in WSNs is critical to overall network performance and even to the energy saving purpose.

CODA [59] is an energy efficient congestion control scheme for WSNs. It comprises three mechanisms: receiver-based congestion detection, open-loop hop-by-hop backpressure and closed-loop multi-source regulation. The congestion detection is based on the combination of the present and past channel loading conditions, and the current buffer occupancy ratio. Backpressure signals will propagate upstream back to the source node right after congestion is detected.

RCRT [40] is a rate-controlled reliable transport protocol, which targets at high-rate and loss-intolerant applications such as imaging, where large volumes of data are generated and flow through the network. RCRT uses end-to-end explicit loss

recovery, but places all the congestion detection and rate adaption functionality at the sink nodes.

The PSFQ protocol presented in [58] is a sink-to-sensors congestion control method. It pumps data segments slowly but fetch data quickly. It provides guaranteed delivery, which is useful for applications such as code updates.

1.3.5 Application Layer

Energy can also be saved at the application layer by reducing data redundancy through in-network processing. A key idea here is to exploit the correlations among the observed data and to remove the data redundancy without loss of useful information to the application. This helps to reduce the network traffic and save energy. In-network processing include compression or aggregation. Data aggregation remains one of the most active research areas in WSNs. An important notion behind data aggregation is that computation is cheaper than communication in terms of energy consumption [25].

1.3.6 Cross Layer

Layering is a software engineering concept. It divides a complex system into smaller modules so that each module has its own requirements and can be handled by different groups of people. People work on one module need little knowledge about other modules. Well-defined interfaces will integrate different modules back into a consolidated system. This divide-and-conquer approach is necessary for building a

complex system. However, due to the lack of knowledge on other layers, an optimal solution found out for one layer may no longer be optimal when things are pieced together. As you may have seen in previous sections, many researches on WSNs have been conducted for finding optimal solutions for energy savings at a particular layer. However, people start to realize that they have to go across the boundaries of existing layers in order to better utilize the scarce resources in WSNs. That is where cross-layer design comes from. Below are some of the examples.

DOZER [9] is a data gathering protocol which meets the requirements of periodic data collection and ultra-low power consumption. It makes MAC-layer, topology control, and routing all coordinated to reduce energy wastage of the communication subsystem. Using a tree-based network structure, packets are reliably routed towards the data sink. Parents thereby schedule precise rendezvous times for all communication with their children. Experiments show that DOZER can achieve radio duty cycles in the magnitude of 0.2%.

[10] proposed a cross layer optimization approach which assumes a very simple MAC protocol and makes use of both routing and MAC layers information to reduce congestion, improve delivery ratio, and optimize energy usage. The proposed approach uses multiple disjoint collection trees, rooted from sink, with non overlapping duty cycles. At the MAC layer, it exploits the fact that nodes that are on different data collection trees need not to communicate with each other, hence the sleep-awake schedules for each tree are different.

[64] proposed a minimum power configuration (MPC) approach to energy conservation in wireless sensor networks. In sharp contrast to earlier research which treats topology control, power-aware routing, and sleep management in isolation, MPC integrates them as a joint optimization problem in which the power configuration of a network consists of a set of active nodes and their transmit powers. Analysis shows that the minimum power configuration of a network is inherently dependent on the data rates at sources. Several approximation algorithms were presented with provable performance bounds compared to the optimal solution, among which is a practical Minimum Power Configuration Protocol (MPCP) that can dynamically (re)configure a network to minimize its energy consumption based on current data rates.

1.4 Organization of the Dissertation

This dissertation is organized in chronological order in which different research topics had been conducted. Chapter 2 presents the Pattern-MAC (PMAC), which is an adaptive energy efficient MAC protocol specifically designed for WSNs. How duty cycles in PMAC adapt to different traffic conditions is explained. Performance of PMAC is evaluated through simulations and comparisons with existing MAC protocols like SMAC and TMAC are made. Chapter 3 presents a software architecture for sensor nodes with dual interfaces. A routing layer component called *switch agent* is introduced to distribute traffic between the two interfaces, depending on traffic

loads. It shows how traffic switching helps in saving energy under light traffic without compromising throughput and end-to-end delay under heavy traffic conditions. Performance is also evaluated through simulations. Chapter 4 presents analytical models for a popular data gathering MAC protocol — DMAC under both CBR traffic and stochastic traffic. The analytical models are evaluated numerically and validated through simulations. Chapter 5 presents a scheme managing sleep schedules at the routing layer. The motivation of this work is explained and comparisons are drawn between the routing sleep scheme and the existing DMAC protocol. Chapter 6 summarizes the work and outlines the direction of my future research.

Chapter 2

An Energy Efficient MAC Protocol for WSNs

2.1 Introduction

Sensor nodes in wireless sensor networks are powered by batteries and are left unattended after deployment. Due to the *ad hoc* nature and the large scale of the network, it is almost impossible to recharge or replace their batteries once they run out of power. Therefore, power saving strategies play a critical role in prolonging the network's lifetime. As you may see from the previous chapter, many research efforts have focused on developing power saving schemes for wireless sensor networks.

Due to the burstiness of traffic in wireless sensor networks, radios on sensor nodes stay in the idle-listening state for most of the time as heavy traffic conditions present only when events occur. It is a known fact that most of radio devices consume as much power in the idle-listening state as in the receiving state. Significant amount of energy will be wasted if radios are left in such idle-listening state. Many energy efficient MAC protocols have been proposed for wireless sensor networks, trying to bring down the *per node idle-listening power consumption* by lowering the duty-cycles of radios through sleep scheduling. In other words, radios are put into sleep

if they are not involved in any data communication. Additionally, it is important to achieve a good balance between energy consumption and performance metrics like throughput and delay. A trade-off among the afore-mentioned parameters can be achieved by adjusting the sleep-awake schedules of sensor nodes. Existing protocols differ in the way they generate the sleep-awake schedules, and consequently yield different trade-offs between energy consumption and performance. Protocols that adapt their sleep-awake schedules to different traffic conditions have been observed to give better performance than the others.

In this chapter, we present a new sensor MAC protocol called Pattern-MAC (PMAC), wherein the actual sleep-awake schedule of a sensor node is determined based on the node's own tentative sleep-awake schedule, and its neighbors' tentative sleep-awake schedules as well. Patterns in the tentative sleep-awake schedules of a sensor node are adaptive to the traffic conditions observed at that node. We find that such a scheme can achieve different degrees of trade-off among energy, throughput and latency, sometimes even better than the existing schemes. In addition, sensor nodes using PMAC exhibit a high degree of *energy localization* when compared to the existing protocols. By energy localization, we refer to the phenomenon wherein only those sensor nodes along the communicating path expend energy, while other nodes in the vicinity do not. Such a feature is indeed a necessity for wireless sensor networks.

Two variations of PMAC are presented in this work — one is in favor of power savings (PMAC-I) and the other is in favor of throughput (PMAC-II). Simulation

results show that both the variants of PMAC exhibit better energy localization when compared to SMAC without adaptive listening [71] and TMAC [56]. Furthermore, in comparison to SMAC without adaptive listening, both the schemes achieve more power savings under light traffic loads, and higher throughput under heavier traffic loads. The PMAC-I also shows better power efficiency[1] when compared with TMAC.

The preliminary results of this work had been published in [73].

2.2 Related Work

Several MAC protocols have been proposed in the literature for sensor networks. The proposed protocols can be classified as either schedule based or contention based [29]. Schedule based protocols use time division multiple access (TDMA) mechanism, wherein each node is assigned a particular time to transmit. Though such protocols offer energy savings by avoiding collisions and idle listening, they have little flexibility in handling traffic fluctuations and node mobility. Contention based protocols do not avoid collisions but can gracefully deal with traffic fluctuations and node mobility. They can be further classified as random protocols and slot based protocols. Random protocols are like "aloha" and allow a node to transmit whenever it wants to. These protocols consume lot of energy in idle listening. While some mechanisms have been proposed to reduce the energy consumption, the efficacy of these mechanisms critically depends on the radio's ability to switch on quickly. Slot

[1]Power efficiency is defined as the ratio of throughput achieved per unit of energy consumed.

based protocols aim to achieve a middle ground between schedule based protocols and random protocols by organizing the sensor nodes in a slotted system.

SMAC [71] is a MAC protocol in which the sensor nodes are synchronized to follow a slotted time structure. Each slot is divided into two periods — awake and sleep. The awake period occurs at the beginning of each slot, during which any node wishing to transmit should contend for the channel. The sleep-awake duration for all the sensor nodes is the same and depends on the duty cycle, which is fixed before hand. Although SMAC saves power, compared to protocols with no duty cycles, it does not adapt to network traffic very well, since its duty cycle is fixed. A duty cycle tuned for heavy traffic loads results in energy wastage when the traffic is light, while the duty cycle tuned for light traffic loads results in low throughput under heavy traffic loads. SMAC with adaptive listening [72] was proposed later on to achieve adaptive duty cycles. In this chapter, we refer to SMAC as the one with fixed duty cycle published in [71]. The Timeout-MAC protocol (TMAC) [56] is another slotted protocol which improves SMAC by introducing an adaptive duty cycle. In TMAC, a node goes to sleep only when there is no activity in its vicinity for a time T_A seconds. Such an adaptation frees the application from the burden of selecting an appropriate duty cycle. TMAC has a similar performance as SMAC under constant traffic loads, but saves more energy under variable traffic.

While TMAC performs better than SMAC, it has poor energy localization characteristics, i.e., even sensor nodes that are not actively communicating expend considerable energy. This is because inactive sensor nodes still need wake up after every

slot time and listen for T_A seconds, to check for network activity. If by some means, a sensor node can get to know that its neighbors will remain inactive over the next several consecutive slots, then it can sleep throughout all those time slots without waking up in between. Such mechanism might result in better energy localization than those in which each node has to wake up after every slot time. This is also the underlying principle of the proposed Pattern-MAC(PMAC) protocol.

2.3 Overview of PMAC

PMAC is a "time slotted" protocol like SMAC, but with a much shorter time slot than SMAC. In SMAC, a node can stay awake for a certain duration of a time slot, and go to sleep for the remaining duration; while in PMAC, a node can either be awake or asleep during a time slot.

2.3.1 Rationale behind PMAC

Idle listening is one of the main sources of energy wastage. Energy saving MAC protocols try to minimize the length of the idle listening period. Fig. 2.1 shows the lengths of idle listening periods in SMAC, TMAC and the proposed PMAC protocols in the extreme case of no traffic in the sensor network.

In SMAC, sensor nodes have to wake up periodically even when there is no traffic in the network, thus wasting power. A small duty cycle can reduce this wastage, but it will cause low throughput when the traffic becomes heavy. In TMAC, sensor

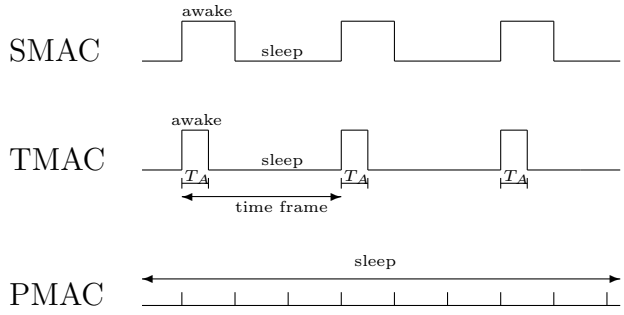


Figure 2.1: Comparison on the length of sleeping periods among SMAC, TMAC and PMAC with no traffic

nodes need to wake up at the beginning of each time frame for a time T_A even when there is no traffic in the network, as a node needs to check for any activity in its neighborhood. In PMAC, a sensor node gets information about the activity in its neighborhood before hand through *patterns*. Based on these patterns, a sensor node can put itself into a long sleep for several time frames when there is no traffic in the network. If there is any activity in the neighborhood, a node will know this through the patterns and will wake up when needed. Thus PMAC tries to save more power than SMAC and TMAC, without compromising on the throughput.

2.3.2 Pattern vs Schedule

A sleep-awake *pattern* is a string of bits indicating the *tentative* sleep-awake plan for a sensor node over several slot times. Bit 1 in the string indicates that the node *intends* to stay awake during a slot time, while 0 indicates that the node *intends* to sleep. For example, a pattern of 001 for a node indicates that, the sensor node

tentatively plans to be asleep for two consecutive slot times, and stay awake in the third. Since the *pattern* is only a tentative plan, it is subject to change.

A sleep-awake *schedule* for a sensor node is a string of bits indicating the *actual* sleep-awake itinerary which the node will follow. Bit 1 in the string indicates that the node *will* stay awake during a slot time, while 0 indicates that the node *will* remain asleep.

The above definitions imply that a node's sleep-awake *pattern* need not be its sleep-awake *schedule*. In PMAC, the *schedule* for a node is derived from its own *pattern* and, the patterns of its neighboring nodes. Therefore patterns do affect the sleep and awake times of a node, and thus the protocol's performance.

In the next few paragraphs, we explain our approach for arriving at a node's pattern and schedule.

2.4 Protocol Details

As explained before, a node's pattern alters its sleep and awake times. In order to achieve a good throughput without compromising on the energy savings, it is important that the generated pattern should adapt to the network traffic.

2.4.1 Pattern Generation

Let P^j be the binary string representing the pattern of a node j. This pattern is associated with node j over N time slots. We call this sequence of N time slots as

a *period*. In case the length of P^j is less than N, then the pattern gets repeated for the remaining duration. For example if $P^j = 01$, and if $N = 5$, then tentative plan for node j over the next five time slots will be 01010, i.e., the node will intend to *sleep* during slots 1, 3, and 5, and to remain *awake* during slots 2 and 4. In PMAC, we restrict a pattern to be 0^m1, where $m = 0, 1, \cdots N-1$. The number of 0 bits in a pattern, denoted by m, indicates the traffic load around the node having the pattern. A large m indicates the traffic load is light, while a small m (even a 0) indicates the traffic load is heavy.

In order to adapt to the traffic conditions, a node's pattern is updated during each period using the local traffic information available at the node and exchanged at the end of each period. Let P_i^j be the working pattern of node j during period i, where $i = 1, 2, \cdots$. Note that P_i^j can be different from P_{i+1}^j depending upon the node j's traffic conditions observed during period i. P_{i+1}^j can be obtained from P_i^j through either single or multiple updates occurring in period i. Let x_i be the number of pattern updates during period i and $P_{i,n}^j$, where $n = 0, 1, \cdots x_i$, be the n^{th} new pattern obtained in the sequence of updates. The starting pattern in the sequence during period i, $P_{i,0}^j$, is the working pattern P_i^j. The last updated pattern in the sequence, P_{i,x_i}^j, is going to be the working pattern in the next period i.e., $P_{i+1}^j = P_{i,x_i}^j$.

When the network is activated, the working pattern at every node has just one bit during the first period, which is 1, i.e., $P_1^j = 1$, $\forall j$ in the network. This simply assumes the traffic load is heavy at the beginning and every node should be awake.

Pattern updates during the first period start with the working pattern P_1^j, i.e., $P_{1,0}^j = P_1^j = 1$. If there is no data[2] for a node j to send at the first time slot of bit 1, then it indicates that the traffic load around node j is potentially light. Therefore, the node can afford to sleep for some time. Hence, node j updates its pattern to 01, i.e., $P_{1,1}^j = 01$. If we find that the node has no data to send during the second time slot of pattern bit 1, the node is encouraged to sleep longer by doubling the number of 0 bits in $P_{1,1}^j$, i.e., $P_{1,2}^j = 001$. This doubling effect continues in the following time slots of bit 1, until the number of 0 bits in the updated pattern reaches a predefined threshold δ. Beyond δ, the number of 0 bits is linearly increased. If there is no data for node j to send during period 1, the following sequence of patterns is generated at node j:

$$1, \quad 01, \quad 0^2\,1, \quad 0^4\,1, \quad \cdots \quad 0^\delta\,1, \quad 0^\delta\,01, \quad 0^\delta\,0^2\,1, \quad 0^\delta\,0^3\,1, \quad \cdots \quad 0^{N-1}\,1.$$

This approach of exponential increasing the sleep time during light traffic allows the nodes to save considerable amount of energy. As you can see from the above, the sleep pattern that is generated mimics the slow-start algorithm of TCP [55].

If node j has any data to transmit at any time slot regardless of the pattern bit at that time slot, then the next pattern in the sequence goes back to 1. This enables node j to wake up quickly to handle the traffic load. The following update, if any, is going to start with this new pattern.

[2]This data can be either the node's own data, or the data generated by other nodes which it has to relay.

The pattern generation scheme used in PMAC is summarized in the following expression:

$$P_{i,n+1}^{j} = \begin{cases} 01 & \text{if } P_{i,n}^{j} = 1 \text{ and} \\ & \text{node } j \text{ has no data to send during the next slot of bit 1;} \\ 0^{2m}1 & \text{if } P_{i,n}^{j} = 0^{m}1 (0 < m \leq \delta/2) \text{ and} \\ & \text{node } j \text{ has no data to send during the next slot of bit 1;} \\ 0^{m+1}1 & \text{if } P_{i,n}^{j} = 0^{m}1 (\delta \leq m < N-1) \text{ and} \\ & \text{node } j \text{ has no data to send during the next slot of bit 1;} \\ 0^{m}1 & \text{if } P_{i,n}^{j} = 0^{m}1 (m = N-1) \text{ and} \\ & \text{node } j \text{ has no data to send during the next slot of bit 1;} \\ 1 & \text{if node } j \text{ has data to send during a slot,} \\ & \text{irrespective of the slot's pattern bit.} \end{cases} \tag{2.1}$$

It is easy to see that by increasing δ, the application can increase the aggressiveness of the sensor nodes to conserve energy. Similar to this multiplicative increase - acute decrease of the sleep times, other schemes such as additive increase - multiple decrease, additive increase - acute decrease, etc. can be employed, if the applications prefer them.

2.4.2 Pattern Exchange

A node's *pattern* is just a tentative sleep-awake plan. In PMAC, the actual sleep-awake *schedule* is derived based on the node's own pattern and the patterns of its neighbors. New patterns that are generated for the subsequent period are broadcasted by the nodes at the end of the current period.

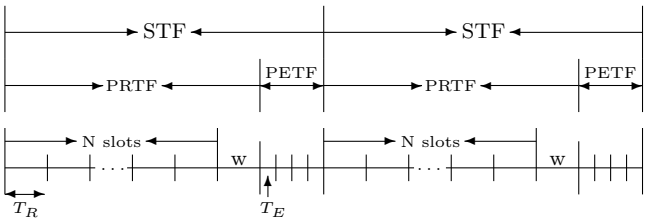


Figure 2.2: Division of time frames

To accommodate this pattern exchange, time is divided into super time frames (STF) as shown in Fig. 2.2. Each STF consists of two sub-frames. The first is called *Pattern Repeat Time Frame* (PRTF), during which each node repeats its current pattern. PRTF in turn is divided into different time slots of duration T_R. PRTF is nothing but the sequence of N time slots that we referred to as a *period* in the previous discussions. At the end of these N slots, PRTF has one additional time slot during which all the sensor nodes stay awake. This special time slot is used to speed up communication. Long delay may happen if the downstream neighbors are in a long sleep mode when upstream nodes have data destined for them. The upstream nodes cannot send data since they know the destination nodes are not ready, while

the downstream nodes might think there is no traffic destined for them, and thus update their patterns for even longer sleep. During this special time slot, data from upstream nodes can be sent to downstream nodes so that downstream nodes can update their patterns to 1 and wake up quickly. This special time slot can also be used for broadcasting.

The second sub-frame of STF is called *Pattern Exchange Time Frame* (PETF), during which new patterns are exchanged between neighbors. PETF again, is divided into various time slots of duration T_E. New patterns are generated during PRTF at every node to reflect the latest traffic information by following the rules summarized in Eqn. (2.1). The last generated pattern during a particular PRTF becomes the pattern for the next PRTF, and will be advertised to the neighbors during the PETF. The pattern is cyclically repeated during PRTF such that each time slot has one pattern bit assigned. Patterns received from its neighbors during the preceding PETF are also repeated in the same way. If a node j receives no new patterns from some of its neighbors during the preceding PETF (probably due to collisions), it then repeats their old patterns.

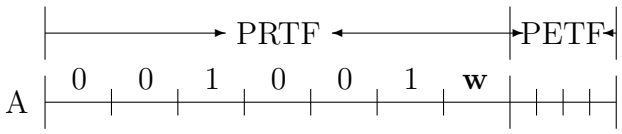


Figure 2.3: Illustration of pattern exchange

Fig. 2.3 illustrates how pattern generation and exchange process takes place in PMAC. In Fig. 2.3, we assume that sensor node A has a pattern 001 at period i. Node A repeats its pattern during PRTF. Let $\delta = 4$ and $N = 6$. Pattern updates start with $P^A_{i,0} = 001$. If node A has no data to transmit during the entire PRTF period, it generates a new pattern, $P^A_{i,1}$, at time slot 3, since the node has pattern bit set to 1 at that time slot. $P^A_{i,1}$ becomes $0^4\,1$ based on the pattern generation rules presented earlier. $P^A_{i,1}$ will be updated at time slot 6, at which another pattern bit 1 appears. $P^A_{i,2}$ now becomes $0^5\,1$ and is the last updated pattern. It is set as the pattern to be exchanged during PETF. However, if there is data at node A at any time slot, the new pattern goes back to 1. For instance, if there is data at time slot 2, $P^A_{i,1}$ is set to 1. Thereafter, if there is no data at time slot 3, $P^A_{i,2} = 01$.

The span of a time slot T_R is chosen such that it is long enough to handle a complete data transmission (contention window + RTS + CTS + DATA + ACK). The choice for N, the number of time slots in PRTF depends on the application. If N is high, then it is possible for the sensor nodes to have more sleep time, and thus more energy can be saved. However, this may also increase the latency in data transmission. Thus there is a tradeoff between energy saving and latency.

The number of time slots in PETF is set to the maximum number of neighbors a sensor node could have. The span of a time slot T_E in PETF is chosen long enough to broadcast a pattern. A large contention window may be needed at the beginning of each PETF time slot to avoid collision. However, longer PETF is, more overheads

are introduced and thus more energy gets wasted. It is a tradeoff between energy saving and reliability.

2.4.3 Schedule Generation

So far, we have explained how a node generates and exchanges its patterns with its neighbors. The purpose of the above exercise is to come up with the actual sleep-awake schedule for a node. To recall, the sleep-awake *schedule* of a node is a string of bits indicating the *actual* sleep-awake itinerary which the node will follow. Each bit in the string indicates the actual state of the node during a slot time. Bit 1 indicates that the node *will* stay awake, while 0 indicates that the node *will* remain asleep.

Table 2.1: Rules to generate the actual schedule at node j in favor of power saving

Pattern bit at node j	Packet to send	Pattern bit at the receiving node	Schedule at node j
1	1	1	1
1	1	0	1−
1	0	$*$	1−
0	1	1	1
0	1	0	0
0	0	$*$	0

1: awake 0: sleep $*$: either 1 or 0

For the first N slots in PRTF, a schedule bit of 1 or 0 is obtained based on the pattern bit values of the node and its neighbors corresponding to that slot. In this chapter, we present two schemes for generating the schedule bits. The first scheme

Table 2.2: Rules to generate the actual schedule at node j in favor of throughput

Pattern bit at node j	Packet to send	Pattern bit at the receiving node	Schedule at node j
1	1	1	1
1	1	0	1
1	0	$*$	1−
0	1	1	1
0	$*$	1 (some neighbor)	1−
0	$*$	0 (all neighbors)	0

1: awake 0: sleep $*$: either 1 or 0

(PMAC-I) is from the receiver's perspective — the sender node can send data only when the receiver is awake. This scheme turns out to be in favor of power saving. The second scheme (PMAC-II) is from the sender's perspective — the sender can send data as long as it is awake and force all potential receivers to be awake. This scheme turns out to be in favor of throughput. The rules associated with these two schemes are summarized in Table 2.1 and Table 2.2, respectively.

The rules for arriving at the schedule bit value for node j for a given slot in PMAC-I are enumerated below:

1. Let the pattern bit at node j be 1 and let there be a packet in its buffer to be sent to a neighbor. If the pattern bit for the receiving node is also 1, then the schedule bit for node j is set to 1. This means that node j will wake up at that particular time slot and send the data, since it knows that the receiver might be awake.

2. Let the pattern bit at node j be 1 and let there be a packet in its buffer to be sent to a neighbor. However, the pattern bit for the receiving node is 0. In this case, the schedule bit at node j to be $1-$, where $1-$ implies node j should wake up at the beginning of that time slot and listen for a certain period of time. If it hears nothing from its neighbor within that period, it can go to sleep. At the first glance, it would appear that it is better to set the actual schedule at node j to 0. That is, let node j sleep from the right beginning of that time slot to save more power. However, since the pattern bit of node j is 1, it could be a potential receiver and its neighbors may try to send data to it. If node j ignores this possible happening, and if it goes to sleep, the packet destined to it will be lost, and the energy spent on transmitting this packet is wasted.

3. The pattern bit at node j is 1 and there is no packet in its buffer. In this case, irrespective of the pattern bits of its neighbors, the schedule bit at node j is set to $1-$. The reason is the same as we explained in case 2.

4. The pattern bit at node j is 0 and there is a packet in its buffer to be sent. If the pattern bit at the receiving node is 1, the schedule bit of node j is set to 1. This implies that, node j is going to wake up at that time slot for transmission, although it intended not to. This can improve the throughput without consuming any additional energy. Throughput is improved by waking

up node j earlier than it is supposed to. No additional energy is consumed because the packet in the buffer needs to be transmitted sooner or later.

5. The pattern bit at node j is 0, and there is a packet in its buffer to be sent. If the pattern bit at the receiving node is 0, the schedule bit for node j is set to 0. This would imply putting node j into sleep, since the destination node is not ready to receive. To send the packet, node j must wait until the time slot at which the destination node has pattern bit 1.

6. The pattern bit at node j is 0 and there is no packet in its buffer. In this case, no matter what pattern bits its neighbors have, the schedule bit of node j is set to 0. This means that node j is going to sleep mode. If some neighbors have packets for node j, they have to wait until the time slot at which the pattern bit of node j becomes 1. This would introduce longer delays for the first few packets when the traffic becomes heavy, but the subsequent packets will experience lower delays as node j's pattern adapts to the new traffic.

In PMAC-II, changes have been made for rules 2,5 and 6 in PMAC-I:

2′) If the pattern bit at node j is 1 and there is a packet to be sent to a neighbor, then the schedule bit for node j is set to 1 instead of 1− even though the pattern bit at the receiver is 0. That means node j can compete for the channel and go ahead to send the data if it wins the channel. By doing this, node j is assuming the receiver is going to be awake to receive the data. From

the perspective of the sender node j, as long as it is awake and has data to send, it can send the data rather than wait for the receiver to become awake. It is obvious that this rule is going to improve the throughput and latency.

5′) Consequently, to guarantee the receiver to be awake when a sender is sending data, rule 5 in PMAC-I must be changed. Let the pattern bit at node j be 0. If one of its neighboring nodes has pattern bit 1, then node j realizes that it could be a potential receiver and thus stays awake for a certain period of time at the beginning of the current time slot. Again this helps to improve the throughput and latency. The side-effect of this approach is that, more nodes may stay awake even when they are not involved in the communication. This results in more power consumption.

6′) The only case, a node j sets it schedule bit to be 0 is when it own pattern bit is 0 and all it neighbors have pattern bits 0 as well.

Based on the application, we can dynamically choose either one of the two schemes to fit the need of the application better.

2.4.4 Channel Access during Wake-up Times

When the nodes are awake, they follow a channel access scheme similar to that of SMAC. The state diagram in Fig. 2.4 shows the transitions from one state to another when a node is awake. At the beginning of each wake-up period, the radio is in idle listening. If a node has data to send, it senses the channel for a randomly chosen

period of time. If channel is free, it sends out *Request To Send* (RTS) and waits for *Clear To Send* (CTS) coming back. After receiving CTS correctly, it sends out data and waits for ACK coming back. Like SMAC, inter frame spaces can be used to allow an ongoing communication to be complete. For those nodes following rule 2 or 3 for setting their schedule bit, they listen for an incoming RTS for a certain amount of time. If no RTS is received or the received RTS is not destined to them, they will sleep for the rest of the current time slot. All nodes sleep and wake up based on their schedules calculated from their patterns.

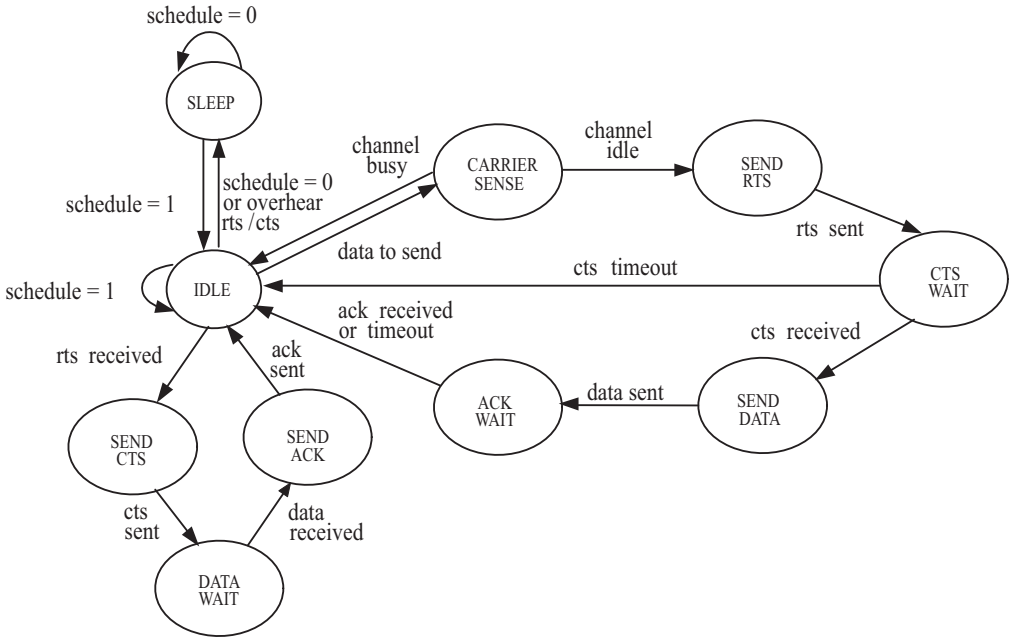


Figure 2.4: State diagram of PMAC

2.4.5 Time Synchronization

Since time is slotted, some level of synchronization among the sensor nodes is needed in PMAC. However, as only large time scales are involved in PMAC (in the order of

hundred milliseconds), small clock drifts would not be a problem. The sensor nodes can use some loose time synchronization schemes, such as those proposed in [57, 35] to synchronize the sensor nodes. Synchronization can be done by introducing a synchronization period at the end of a PETF.

2.5 Qualitative Discussion

In this section, we give a qualitative discussion on the efficacy of PMAC.

2.5.1 Adaptability to Traffic Conditions

As we stated in the previous section, the number of 0 bits in a new pattern grows exponentially when the traffic load is light. This means that sensor nodes can fall into a long sleep quickly under light loads. Hence, PMAC is able to save more power than SMAC. If any data is detected during the current PRTF, the new pattern generation process will start over from 1. This enables, a sensor node to wake up quickly when the traffic load becomes heavy. PMAC is thus able to adapt to the traffic conditions. We also note that the pattern repeating process may compromise the speed of adapting to the network traffic, since new patterns must wait until the next PRTF to be effective.

2.5.2 PMAC-I *v.s.* PMAC-II

Two variants of PMAC, PMAC-I and PMAC-II, have been presented to address the tradeoff between power saving and throughput. PMAC-I is designed from a

receiver's perspective — if a receiver is sleeping when a sender has data for it, the sender has to wait until the receiver wakes up. This approach is going to conserve lots of energy, but the throughput may not be high. PMAC-II is designed from a sender's perspective — all potential receivers are forced to stay awake for a period of time such that the sender can send data as long as it is awake. This approach gives higher throughput, since the sender do not have to wait until the receiver wakes up. However, since all the potential receivers are forced to stay awake for some time, energy will be wasted at those nodes which are not the actual receiver.

PMAC-I adapts to traffic condition better in terms of power saving and energy localization, while PMAC-II adapts to traffic condition better in terms of throughput and latency. PMAC-I can be used in cases where the energy saving is the major concern, while PMAC-II can be used in cases where the throughput and latency become major concerns. We can also use both schemes in a single application at the same time and dynamically choose the one which fits the needs of the application better to achieve good balance between energy saving and network performance.

2.5.3 Power Savings through Localization

In PMAC, only those sensor nodes involved in a communication will wake up frequently. Other sensor nodes that do not participate in the data gathering/relaying process will sleep for longer times. In other words, PMAC selectively wakes up sensor nodes. Fig. 2.5 illustrates the power saving through localization feature of PMAC on a 5×5 mesh. Suppose that the only traffic in the network is from the source

node to the sink node along the path indicated by the arrows. At steady state, only those sensor nodes on the path will stay awake to handle the traffic. Other nodes not involved in the communication will sleep through most of the PRTF period.

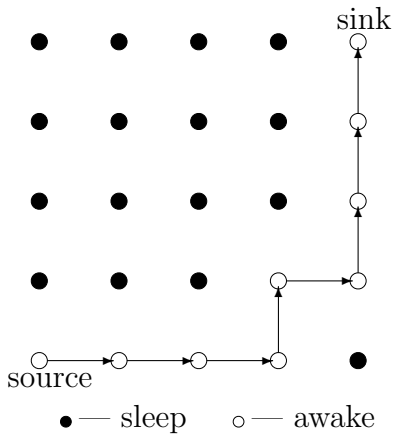


Figure 2.5: An example topology to illustrate energy localization

2.5.4 Power Savings through Reduced Idle Listening

We have just described how PMAC saves energy by allowing sensor nodes that are not involved in any communication to remain asleep. This in turn, reduces the energy wasted due to idle listening during the periodic wake-ups that take place in SMAC. PMAC can also potentially introduce additional idle listening than SMAC. This occurs if in the actual schedule of a sensor node, there are two consecutive awake time slots, but during the second time slot no communication is associated with the sensor node. This is the case where its pattern has two consecutive 1's when the traffic load is light. Fortunately, because of the sparse spurts of traffic in sensor networks, this kind of pattern does not occur quite often.

2.5.5 Resource Usage

The patterns followed by tentative schedules are just binary strings, which can be implemented using bitmaps. Even though PMAC stores the patterns of its neighbors, the extra memory usage incurred should be negligible. The pattern evolution process and actual schedule generation can be done through bitwise operations, which requires insignificant cpu time. The pattern exchange is an overhead in terms of energy consumption, compared to existing protocols like SMAC and TMAC. However, the memory usage and cpu time for pattern exchange are not significant either. In a word, PMAC does not introduce significant resource usage in terms of memory and cpu time compared to existing protocols like SMAC or TMAC.

2.6 Analytical Model

We present a simple analytical model to study the pattern generation process and calculate the steady state average power savings in PMAC under light traffic. We calculate the average power savings by estimating the number of zero bits appearing in the pattern. For simplicity, we ignore the pattern repetition during PRTF presented earlier, and study PMAC with the following model. Starting with a working pattern 1, if no data is available at a node for transmission at the time slot with bit 1, the pattern for the next two slots is set to be 01. Again, if no data to be sent during the previous two time slots, the next working pattern for the subsequent slots becomes 001, and so on. While the above model may not completely capture

the complete traits of the protocol, the analysis does provide good insight into the results that follow.

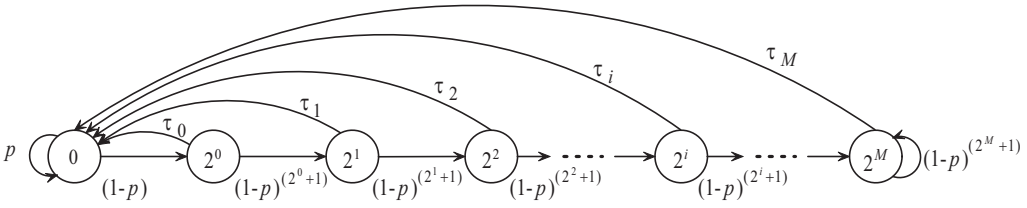


Figure 2.6: Markov chain of the pattern generation process

Let p be the steady state probability that the buffer at node j is non-empty at a particular time slot. As we are interested in the steady states, it is reasonable to assume p is a constant over all the time slots. For the sake of simplicity, we also assume $\delta = N = 2^M$. Now we can use the Markov chain as shown in Fig. 2.6 to model the pattern evolution process. In the figure, each state is indicated by the number of 0 bits in its pattern. Let $P_0, P_{2^0}, P_{2^1}, P_{2^2}, \cdots, P_{2^i}, \cdots, P_{2^M}$ be the probability that the sensor node is in each of those states. We can have the following equations based on the Markov chain:

$$\begin{aligned} P_{2^0} &= (1-p)P_0 \\ P_{2^1} &= (1-p)^{(2^0+1)}P_{2^0} \\ P_{2^2} &= (1-p)^{(2^1+1)}P_{2^1} \\ &\vdots \quad\quad \vdots \\ P_{2^i} &= (1-p)^{(2^{i-1}+1)}P_{2^{i-1}} \end{aligned}$$

$$\begin{aligned} \vdots \qquad & \vdots \\ P_{2^M} &= (1-p)^{(2^{M-1}+1)} P_{2^{M-1}} + (1-p)^{(2^M+1)} P_{2^M} \end{aligned}$$

By substitution, we have for $0 \leq i \leq M-1$

$$\begin{aligned} P_{2^i} &= (1-p)^{(2^{i-1}+1)} \cdots (1-p)^{(2^0+1)} (1-p) P_0 \\ &= P_0 (1-p)^{\left[\left(\sum_{j=0}^{i-1} 2^j\right)+(i+1)\right]} \\ &= P_0 (1-p)^{(2^i+i)} \end{aligned} \tag{2.2}$$

and

$$\begin{aligned} P_{2^M} &= (1-p)^{(2^{M-1}+1)} P_0 (1-p)^{(2^{M-1}+M-1)} + (1-p)^{(2^M+1)} P_{2^M} \\ &= \frac{P_0 (1-p)^{(2^M+M)}}{1-(1-p)^{(2^M+1)}} \end{aligned} \tag{2.3}$$

From the Markov chain, we should also have

$$P_0 = pP_0 + \tau_0 P_{2^0} + \tau_1 P_{2^1} + \cdots + \tau_i P_{2^i} + \cdots + \tau_M P_{2^M} \tag{2.4}$$

where $\tau_i = 1-(1-p)^{2^i+1}$ is the transition probability from state P_{2^i} to P_0 for all $0 \leq i \leq M$. We can verify that Eqn. (2.4) holds by plugging Eqn. (2.2) and Eqn. (2.3)

into it. Since a sensor node must be at one of those states, the summation of the probabilities should be 1.

$$\begin{aligned} 1 &= P_0 + P_{2^0} + P_{2^1} + P_{2^2} + \cdots + P_{2^i} + \cdots + P_{2^M} \\ &= P_0 + P_0 \sum_{i=0}^{M-1} (1-p)^{(2^i+i)} + P_0 \frac{(1-p)^{(2^M+M)}}{1-(1-p)^{(2^M+1)}} \end{aligned} \tag{2.5}$$

Hence,

$$P_0 = \frac{1}{1 + \sum_{i=0}^{M-1} (1-p)^{(2^i+i)} + \frac{(1-p)^{(2^M+M)}}{1-(1-p)^{(2^M+1)}}} \tag{2.6}$$

Now the probability at Eqn. (2.2) becomes

$$P_{2^i} = \frac{(1-p)^{(2^i+i)}}{1 + \sum_{i=0}^{M-1} (1-p)^{(2^i+i)} + \frac{(1-p)^{(2^M+M)}}{1-(1-p)^{(2^M+1)}}} \tag{2.7}$$

for $0 \leq i \leq M-1$. The probability at Eqn. (2.3) becomes

$$P_{2^M} = \frac{1}{1 + \sum_{i=0}^{M-1} \frac{1-(1-p)^{(2^M+1)}}{(1-p)^{(2^M-2^i+M-i)}} + \frac{1-(1-p)^{(2^M+1)}}{(1-p)^{(2^M+M)}}} \tag{2.8}$$

The average number of 0 bits, denoted by $E(0)$ in a pattern, can then be obtained as follows

$$E(0) = \sum_{i=0}^{M} \left(2^i \, P_{2^i}\right) \tag{2.9}$$

When the traffic is heavy, p is close to 1. From Eqn. (2.6), we can see that P_0 is also close to 1 and thus $E(0)$ is close to 0. When the traffic is light, p is close to 0. From Eqn. (2.8), we can see that P_{2^M} tends to 1 and thus $E(0)$ is close to 2^M.

We will now derive an expression to determine the additional amount of power saved at a particular node in PMAC over SMAC. Consider a time interval of length $E(0) * T_R$, where T_R is the slot time in PMAC. It is easy to see that, over this entire duration, a sensor node will be asleep in PMAC. If T is the frame duration in SMAC and d is the duty cycle, then over the same time interval of $E(0) * T_R$, a node in SMAC will be awake for the duration $\frac{E(0)*T_R}{T} * d$. Thus the amount of additional energy saved at a particular node in PMAC over SMAC is given by

$$E_{save} = \frac{E(0) * T_R * d * P_{idle}}{T} \tag{2.10}$$

where P_{idle} is the power consumption when sensor nodes are in idle listening state. Let S be the number of sensor nodes in the network not involved in the data gathering/relaying process, then the idle listening energy saved by PMAC in the sensor network during the time interval can be calculated as

$$E_{save}^{total} = S * \frac{E(0) * T_R * d * P_{idle}}{T} \tag{2.11}$$

2.7 Simulation Results

We have simulated SMAC, TMAC and PMAC(I and II) using the latest version of *NS-2*. The energy consumption, throughput and latency of those three protocols have been investigated and comparisons have been made.

Table 2.3: Parameters used in the simulations

Parameter	Value
initial energy	100 Joules
transmit power	24.75 mW
receiving power	13.5 mW
idle power	13.5 mW
bandwidth	20 kbps
contention window	63 ms

The simulations were done on a 5×5 mesh topology, as shown in Fig. 2.5. In all the simulations, we have used UDP as the transport layer protocol and *variable bit rate* traffic sources with exponentially distributed "on" and "off" periods are used. *Constant bit rate* sources are applied during each "on" period. Different traffic loads are achieved by changing the traffic rate during the "on" periods. The simulation time is set to 1500 seconds. For PMAC, $T_R = 258$ms and $T_E = 104$ms, where T_R and T_E are the slot time in PRTF and PETF, respectively. The number of time slots in PRTF is 64 and the number of time slots in PETF is 4. For SMAC, the listen time is set to 143ms, the sleep time 1290ms and the cycle time 1433ms. The same cycle time is used for TMAC. The T_A time in TMAC is set to 142ms. The same

virtual cluster scheme as for SMAC is used for TMAC in the simulation. Some of the parameters used in the simulations are listed in Table 2.3, wherein we used the same radio parameters as presented in the SMAC paper [71]. Those parameters are based on the low power radio transceiver module TR1000 from RF monolithics, Inc.

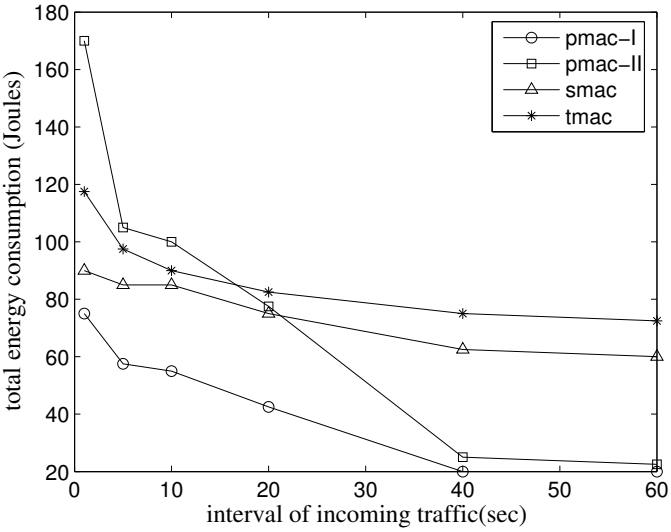


Figure 2.7: Comparison on total energy consumption among SMAC, TMAC and PMAC under different traffic loads

Fig. 2.7-2.9 compare the energy consumption, throughput and power efficiency among SMAC, TMAC and PMAC(I and II) under different traffic loads with varying time intervals: 1, 5, 10, 20, 40, 60 seconds. This simulation was done on the same 5×5 mesh network as shown in Fig. 2.5. Fig. 2.7 shows that PMAC-I consumes least energy among those four approaches for all traffic loads. The energy consumption in PMAC-I and PMAC-II drop much faster than SMAC and TMAC as the traffic becomes light. That means both PMAC-I and PMAC-II can adapt to the traffic better than the other two schemes in terms of power saving. Although TMAC adopts an adaptive duty cycle, it still consumes more power than expected. The

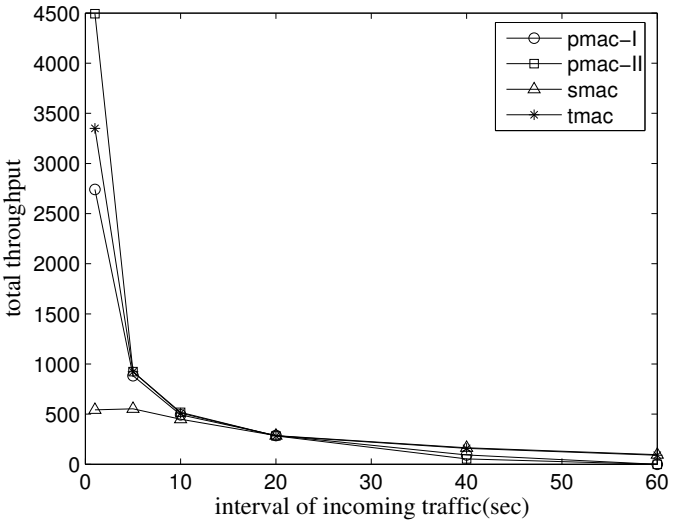


Figure 2.8: Comparison on total throughput among SMAC, TMAC and PMAC under different traffic loads

reason could be that the TA timers may be scheduled multiple times and thus keep nodes idle listening for a long time. Fig. 2.8 shows that the throughput of all the four protocols are pretty close when the traffic is light, indicating sleeping does not affect traffic flowing through in this case. However, when the traffic is heavy, TMAC, PMAC-I and PMAC-II all have significant improvements on the throughput. That is because all the three protocols are using adaptive duty cycles, while periodic sleeping due to fixed duty cycle blocks the traffic flowing through in SMAC. PMAC-II can have better throughput than TMAC although PMAC-I can not. Fig. 2.9 compares the four protocols in terms of their power efficiency. Power efficiency, which is the throughput achieved per unit of energy consumed, is given as

$$\text{power efficiency} = \frac{\text{total throughput}}{\text{total energy consumption}}$$

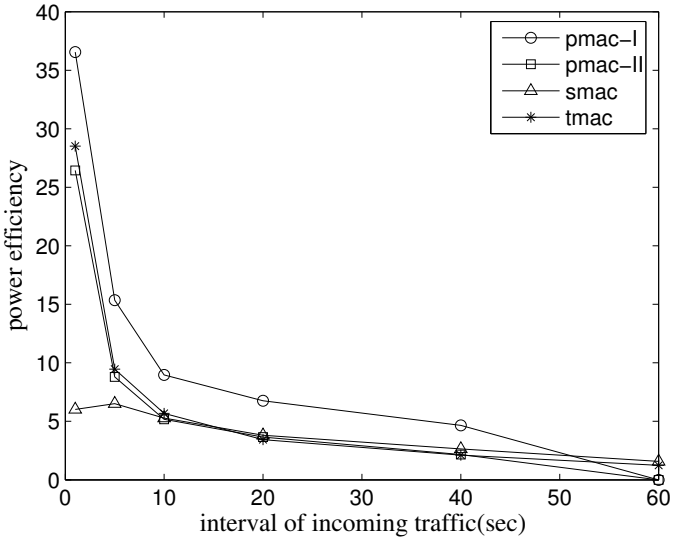
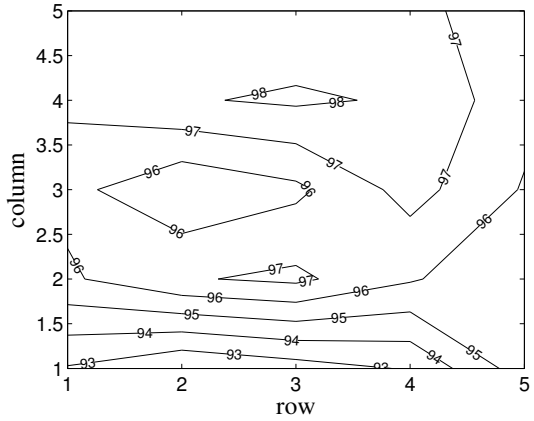


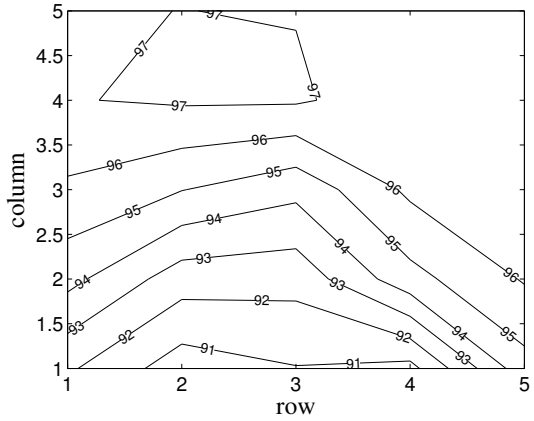
Figure 2.9: Comparison on power efficiency among SMAC, TMAC and PMAC under different traffic loads

It has been observed that all the three protocols with adaptive duty cycles have better power efficiency than SMAC, especially when the traffic load is heavy. PMAC-I has the highest power efficiency although its throughput is not as high as TMAC and PMAC-II. That also means PMAC-I is the best choice among those four protocols when the energy saving is the major concern.

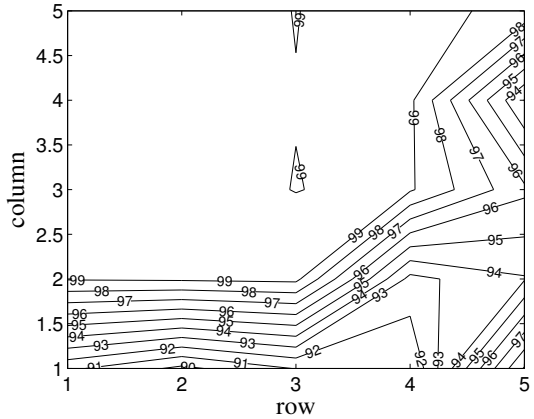
Fig. 2.10 shows the contour maps of the residual energy distribution of a 5×5 mesh network with the same communication path as shown in Fig. 2.5. We observe that in PMAC-I, those nodes at the upper-left corner and the lower-right corner, which are not involved in the communication, have more residual energy in comparison with the other three. This is because in SMAC, those nodes not involved in the communication still have to wake up periodically; in TMAC, the TA timer could be scheduled multiple times and thus keep nodes idle-listening for a long time even



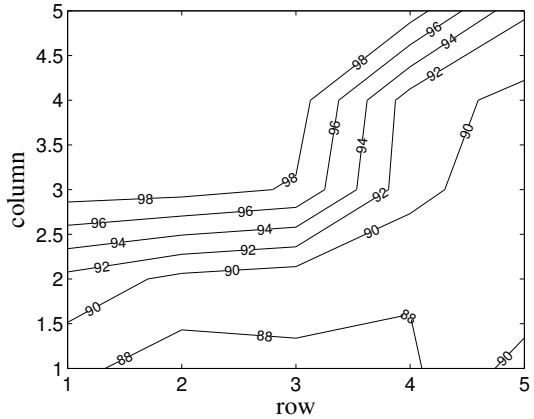
(a) Residual energy distribution on a 5×5 mesh for SMAC.



(b) Residual energy distribution on a 5×5 mesh for TMAC.



(c) Residual energy distribution on a 5×5 mesh for PMAC-I.

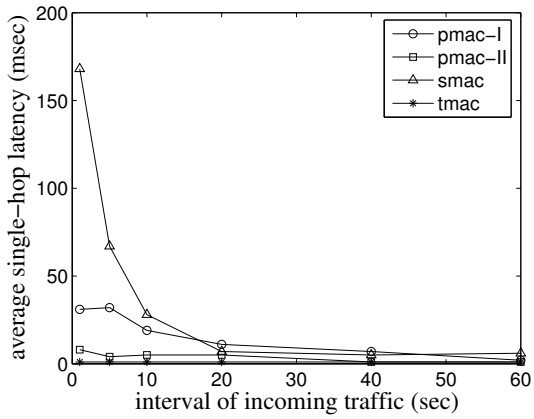


(d) Residual energy distribution on a 5×5 mesh for PMAC-II.

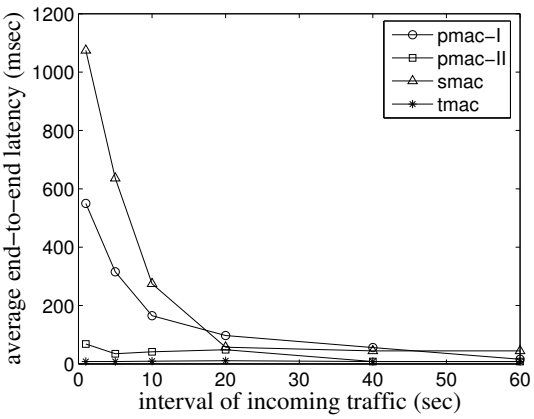
Figure 2.10: Contour maps of residual energy on a 5×5 mesh for SMAC, TMAC and PMAC

not involved in a communication; in PMAC-II, according to rule 5, all the potential receivers must stay awake for a certain period of time in favor of throughput, hence some nodes waste energy unnecessarily. In a word, PMAC-I shows better energy localization than the other three.

Fig. 2.11 shows the single-hop and end-to-end latency of all the four protocols. Both Fig. 2.11(a) and Fig. 2.11(b) show that SMAC has the highest single-hop and



(a) Comparison on average single-hop latency among SMAC, TMAC and PMAC under different traffic loads



(b) Comparison on average end-to-end latency among SMAC, TMAC and PMAC under different traffic loads

Figure 2.11: Comparison on latency among SMAC, TMAC and PMAC under different traffic loads

end-to-end latency. That is because the periodic sleeping blocks the traffic flow through fast. PMAC-I also has pretty high latency. That is the sender can not send data if the receiver has pattern bit 0 until the pattern bit at the receiver is changed. This may be delayed until the next PRTF and thus causes a high latency. Fortunately, PMAC-II can achieve latency as low as TMAC. That is because it is more in favor of throughput and the sender does not need wait for the receiver to become awake. Obviously, TMAC and PMAC-II are better choices when the throughput and latency become major concern.

2.8 Summary

This chapter presents an energy efficient MAC protocol, called PMAC, where patterns in tentative sleep-awake schedules of a sensor node are adaptive to the traffic

conditions observed by that node. Patterns are exchanged among neighbors after some time. The actual sleep-awake schedules are generated based on a sensor node's own patterns and its neighbors' patterns. Our simulation results show that in comparison to SMAC, PMAC achieves more power savings under light loads, and higher throughput under heavier traffic loads. The PMAC-I also show better power efficiency when compared with TMAC. The improved performance of PMAC indicates that "pattern exchange" is a promising framework for improving the energy efficiency of the MAC protocols used in sensor networks.

Chapter 3

A Traffic-Aware Switch Agent for WSNs

3.1 Introduction

There have been a number of approaches to deal with the bursty traffic in wireless sensor networks with sensor nodes bearing a single radio interface. These approaches include protocols at both medium access control (MAC) and routing layers. Adaptive MAC protocols such as SMAC [71], TMAC [56] and PMAC [73] adopt adaptive duty cycles based on the traffic load. The radio is woken up when the traffic load is heavy and is put into sleep when the traffic load is light. Those protocols assume the physical layer has a large enough bandwidth to handle the peak traffic.

Dual radios can also be used to deal with bursty traffic. Dual radios typically consist of one radio with a relatively high bandwidth, high power and longer transmission range, and the other with a relatively low bandwidth, low power and shorter transmission range. The low-bandwidth and low-power radio is used under light traffic condition to save energy, while the high-bandwidth radio is turned on only under heavy traffic conditions. In this chapter, we present and evaluate the architecture of a distributed routing-layer switch agent for wireless sensor nodes equipped with

such kind of dual radios. The switch agent intelligently activates the high-bandwidth and high-power radio based on either traffic condition or end-to-end delay. Compared with existing works, our approach is completely decentralized in that each node makes a local decision to wake up the appropriate high-bandwidth radios for transferring the data to the destination. Each radio interface has its own routing agent so that a better path can be found for the high-bandwidth radios. A switch agent sits on top of the two routing agents to distribute traffic between the dual radio interfaces. We have also proposed supporting enhancements (such as schemes for maintaining the routing cache) that can result in additional energy savings at network nodes. It is of practical value as a distributed and energy efficient way to handle bursty traffic in wireless sensor networks with dual radios.

We have provided extensive simulation results to test the performance of our proposed switch agent architecture using *NS-2*. Our simulation results indicate that: (i) the end-to-end delay and throughput achieved by the proposed distributed interface switch framework are comparable to those achieved in a network of sensor nodes equipped only with IEEE 802.11 radios, (ii) the energy consumed in the network using our interface switch framework is a fraction of that consumed in a network of the IEEE 802.11 sensor nodes and is quite comparable to that of sensors using only IEEE 802.15.4 radios.

The preliminary results of this work had been published in [74].

3.2 Related Work

A few previous work had been done on dual radios. [43] proposed the CoolSpots model which focuses on using bluetooth and WiFi radios in a single hop mode. A few switching policies have been proposed to make the switching decisions. Although reference [43] claims that the CoolSpots model can be used for *ad hoc* peer-to-peer configuration, it does not address the complexities of the routing layer that arise as a result of different transmission ranges of the two radios. [70] proposed a network architecture consisting of a set of high-bandwidth nodes and low bandwidth nodes. The low bandwidth nodes connect to the high-bandwidth nodes thereby reduce the total number of transmissions to reach the destination. This increases the lifetime of low-bandwidth nodes. The high bandwidth nodes are assumed to be connected to a power source and hence do not have any energy constraints. [60] proposed the usage of dual radios to avoid network congestion and maintain the application fidelity in overload traffic conditions. A small number of statically placed virtual sinks with separate longer-range radios form a secondary network. The longer-range radios at the virtual sinks stay active all the time. Traffic will be redirected to the secondary network when network congestion is detected in the primary network. When both the primary and secondary networks are overloaded, the system falls back on the traditional congestion mitigation schemes like rate control or packet dropping. The drawbacks of this approach are that statically placed virtual sinks may not adapt to network conditions very well and there is no power management scheme adopted for

the longer-range radios. In this research, the longer-range radios are dynamically selected and activated only when needed, which saves energy since overload traffic conditions only account for a small portion of the network lifetime in most cases.

A more recent work by [52] introduces ideas similar to the ones introduced by us in this research. They proposed a network architecture containing devices that have a single low-bandwidth radio and devices with both low-bandwidth and high-bandwidth radios. The low-bandwidth radios are always turned on and the high-bandwidth radios are always turned off. However, the high power radio at a specific node called the topology controller is always kept on. When a low-bandwidth node has data to send to a particular destination, it sends a path request to the topology controller. The topology controller selectively wakes up the high-bandwidth radios along the path from the source to the destination. The data travels along low-bandwidth nodes to high-bandwidth node, passes through a sequence of high-bandwidth nodes, and finally goes from the high-bandwidth node to the destination along low-bandwidth nodes. While the work done in [52] is interesting, it has a few drawbacks. It uses a centralized controller (with no energy constraints) which partially negates the advantages and philosophy of a truly distributed sensor network. Also, only the source nodes are allowed to make the decision on using the high-bandwidth nodes for transmission which could decrease the utility of the high power radios.

3.3 Overview of the Switch Agent

We assume the high-bandwidth radio has a relatively longer transmission range than the low-bandwidth radio. This is true for radios used with the IEEE 802.11 and the IEEE 802.15.4, even though high-bandwidth radio does not necessarily assume a longer range, for instance, the ultra-wide band radios in the IEEE 802.15.3a or the wireless USB devices have high bandwidth but fairly short transmission range.

We will call the interface with the lower bandwidth and shorter transmission range as Interface-I and the one with the higher bandwidth and longer transmission range as Interface-II. A switch agent is a software component that distributes traffic between these two interfaces. By default, Interface-II is powered-off and is woken up by sending appropriate control message along Interface-I. We will assume that Interface-I is always active (with appropriate duty cycle) and that the network is connected when all the nodes activate their Interface-I. A distributed protocol such as AODV [44] is executed to populate the routing tables at each node. This routing table will provide next-hop information along different routes in the network based on Interface-I. Since the reachability of Interface-II is higher than Interface-I, at each sensor node we need to find a new set of routing table based on Interface-II. If the reachability of Interface-II is a multiple of the reachability of Interface-I, then with appropriate modification to the AODV protocol, it may be possible to approximate the routing table related to Interface-II from the Interface-I routing tables, thus saving energy at nodes. In general, this may not be possible. Additionally, running

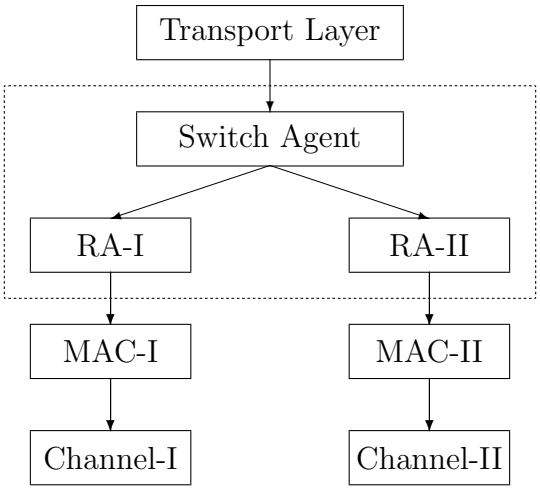


Figure 3.1: Positioning of the switch agent

the routing protocols on Interface-II will likely find a shorter higher-bandwidth path than constructing one from the Interface-I routing tables.

In order to use a different routing agent for the Interface-II, the switch agent is placed above the routing agents. The protocol stack is illustrated in Fig. 3.1, where RA-I and RA-II are the routing agents used by Interface-I and Interface-II, respectively.

Two switching schemes have been proposed for the switch agent based on different application delivery requirements: the queue-length switching and the delay-bound switching. In the queue-length switching, switching occurs whenever an interface queue length exceeds a predefined threshold. In the delay-bound switching, switching occurs whenever the end-to-end delay of a certain traffic flow exceeds an upper-bound limit demanded by applications. The queue-length switching is in favor of an

application's throughput demands, while the delay-bound switching is in favor of an application's demands on end-to-end delays.

3.3.1 Components of the Switch Agent

Every switch agent has the following components: interface queue monitor for queue-length switching, end-to-end delay monitor for delay-bound switching, sleep-wakeup unit, route cache unit, and timers.

3.3.1.1 Interface Queue Monitor

The duty of the switch agent is to switch Interface-II on and forward the traffic to Interface-II in order to meet the application demands and/or when the traffic rate at a node becomes high, necessitating the use of Interface-II. In order to make this happen, we need a component to monitor the packet transmission queue at every node. In this research, we have chosen to monitor the length of the interface queue in between RA-I and MAC-I, since the queue length reflects the cumulative effect of both incoming traffic rate and transmission rate. We will use a predefined threshold called THRESHOLD_HT. Whenever the queue length exceeds THRESHOLD_HT and if the Interface-II is on, then the incoming traffic is diverted to Interface-II.

3.3.1.2 End-to-end Delay Monitor

For delay-bound switching, the switch agent at every node x maintains a data structure. For a traffic flowing to a destination d through this node x, there will be an

entry for it in the data structure. Let us assume node y is the previous hop of this node x, if x is not the source node where the traffic flow is initiated. Then the entry can be expressed as a quadruple [d, D_e, D_l, T_e], where d denotes the destination node id, D_e denotes the last updated end-to-end delay from this node x to the destination node d, D_l denotes the last updated one-hop delay from the previous hop y to this node x, and T_e denotes the expiration time at which the entry will be removed. Initially, both D_e and D_l have a value of 0. Whenever node x receives a data packet through its Interface-I, it will find out the current one-hop delay D_c of the data packet by computing the difference between the current time and the time stamp (when entering the previous hop y) in the packet header. Note that every data packet is time-stamped when entering a node. If node x is the destination node or an intermediate node other than the source node in which an entry already exists for the same destination d, a comparison between the current one-hop delay D_c and the last updated one-hop delay D_l will be drawn. If the difference exceeds a predefined threshold DIFF, then a delay update will be triggered. This node x will add the current one-hop delay D_c into its last updated end-to-end delay D_e and send the result back to its previous hop y. Meanwhile, the current one-hop delay D_c will replace the last updated one-hop delay D_l. Once the previous hop y receives the delay packet, it will update its end-to-end delay. It turns out the delay updates are initiated at destination nodes and propagate back to source nodes hop by hop. The reason of having DIFF is to limit the number of delay packets flowing around in the networks so that the delay packets would not use too much bandwidth.

3.3.1.3 Route Cache

On-demand routing agents like AODV, establish routes whenever there are data packets bound for a certain destination. We will use two route caches, one for each interface. We will denote these route caches (the routing tables) as RC-I and RC-II corresponding to Interface-I and Interface-II, respectively. When the network initially starts up, only Interface-I is on. If there is a data packet originated at some node, RA-I at that node will broadcast a route request. A route will be established after receiving a route reply from either the destination node or a node knowing how to reach the destination node. The routing agent like AODV can reestablish a route whenever a route is expired or repair a route when the next-hop neighbor is dead. All the Interface-IIs will be turned on for the initial switching since no cached routes yet. RA-II will do the same to establish a route and the route will be cached for the subsequent switching.

A RC-I entry is made to expire if messages along Interface-I are not delivered to the next-hop specified in that entry. In this case, the entry is purged and the routing agent at the node is activated to recalculate the route. An RC-II cache entry (corresponding to Interface-II) will expire if it has not routed any data packets on it for a period of time. Apart from caching all the route information to destinations at relating to Interface-I, the number of hops (with Interface-I) required to reach each destination can also be stored and updated from time to time. The caching of hop counts will help to make a ringcast instead of broadcast for sending wake-up control

messages. Ringcast is a broadcast but restricting the TTL (Time-To-Live) of the broadcasted packet to a limited number of hops. This can save energy and improve the scalability of the protocol.

3.3.1.4 Sleep-wakeup Unit

The sleep-wakeup unit at a node is responsible for (i) turning on Interface-II at the node and (ii) sending control messages along the node's Interface-I to turn on Interface-II at other nodes. If an entry for a particular destination has to be populated in RC-II, we have resort to a broadcast/ringcast to determine the path (made up of Interface-II) to the destination. These broadcast/ringcast control messages are sent by the sleep-wakeup unit along Interface-I. Let us consider a scenario when a RC-I entry is available and the RC-II entry has expired at node x for destination d. The cache RC-I at x will give us the next hop node (say y) on the path to d using Interface-I. Waking up node y's Interface-II may not be prudent since we can possibly bypass node y owing to the increased range of Interface-II at x. For this reason, node x will send a broadcast/ringcast message to wake up Interface-II on all the nodes that receive the message. After this, routing protocol is run on Interface-II to create an entry for d in RC-II after which x starts forwarding data along this path. The nodes that turned on their Interface-II will turn it off in case they do not receive any data for forwarding along Interface-II for a period of time.

Now consider the scenario that both RC-I and RC-II entries are available at node x for destination d. Having a RC-II entry for d does not imply that the route

is active, since the intermediate nodes along that path could have turned off their Interface-II due to inactivity. Let z be the next hop neighbor of x on RC-II towards d. Further let y' be the next hop node in RC-I to the destination z. Now a unicast wake-up control message is sent to node y' using Interface-I at node x with the destination as node z.

A wake up registry is maintained at node x to indicate that node z is woken up to route data packets to destination d. This will avoid unnecessary transmission of wake-up messages. The entries in the registry are purged from time to time.

3.3.1.5 Timers

Three timers are maintained in the switch agent: IDLE_TIMER, CACHE_TIMER and REGISTRY_TIMER. The IDLE_TIMER keeps track of the traffic that is seen by Interface-II. In the absence of any traffic for a duration of time, defined by IDLE_INTERVAL, Interface-II will be turned off. CACHE_TIMER is fired periodically to purge old paths and determine new ones. The REGISTRY_TIMER is maintained to purge the entries in the registry. Each registry entry will be of the form $[d, t]$, where d is the destination and t is the timestamp indicating the latest time the node has seen a data traffic to destination d through its Interface-II. When the REGISTRY_TIMER is fired, all the entries that satisfy the relation $curTime - t >$ REG_TTL will be removed, where $curTime$ is the current wall clock time and REG_TTL is a predefined threshold.

3.4 Protocol Details

As stated earlier, the switch agent sits at the routing layer and on top of two routing agents, each of which is for two different interfaces. In our distributed protocol nodes receive four types of packets:

- wake-up packets - control packets generated from the switch agent to wake up the Interface-II. Wake-up message could be either a unicast message or a broadcast/ringcast message. All control packets communicate through the Interface-I;
- routing packets - control packets generated by the routing agents to establish or update a route;
- delay packets - control packets for updating end-to-end delays. Used only for the delay-bound switching scheme.
- data packets - data packets originated from the sensors.

In each of the intermediate nodes along a route, every packet must go through all the protocol layers until the switch agent. When the data packet reaches its destination node, it will be passed to the upper layers above the routing layer.

We tag all packet headers with a channel ID. The data packet generated by the application or the control packet generated by the switch agent at the node of origin will have a channel ID 0. The switch agent, after determining the channel to use, will use an ID 1 and 2 for the packets going through Interface-I and Interface-II,

respectively. We will describe the protocol details based on the types of the packet the switch agent receives.

3.4.1 Receiving a Routing Control Packet

When the switch agent receives a routing control packet, it will first check the header of the packet. If the header is tagged with a channel ID 1 (resp. 2), the routing packet will be forwarded to RA-I (resp. RA-II).

3.4.2 Receiving a Delay Packet

For the delay-bound switching scheme, whenever the switch agent at a node receives a delay packet, it first check if there exists a delay entry on the list for a particular destination. If not, add a new entry. Then simply update the end-to-end delay in the existing or the newly added entry.

3.4.3 Receiving a Wake-up Packet

A wake-up packet sent by a node could be either a broadcast/ringcast message or a unicast depending upon the availability of caching information.

1. Receiving a broadcast/ringcast message: If the switch agent receives a broadcast/ringcast message, it will drop the packet when TTL is 0. If the TTL is greater than zero, the node's sequence number will be used to eliminate broadcasting of the old packet. In response to the new wake-up packet, Interface-II will be started. If this particular node does not observe data flowing through

its interface for a period of time IDLE_INTERVAL after it has been switched on, then its Interface-II will be put into sleep. An IDLE_TIMER will be used to countdown this interval of time.

2. Receiving a unicast message: If the switch agent receives a unicast wake-up message through Interface-I and it is the next-hop identified, then the Interface-II will be switched on and the IDLE_TIMER will be rescheduled. Now it will determine if a unicast or broadcast/ringcast message has to be sent to wake up the nodes along the path to the destination. This decision is based on information available at the cache RC-II. In addition, before sending a unicast message is the wakeup registry searched to see if the next hop node is already active on Interface-II. If it is active on Interface-II, then the wakeup message along Interface-I is avoided and the data packets are sent via Interface-II to the next hop node directly.

   If the node receiving the wake-up message is not the next-hop that is identified, then the message will propagate towards the next-hop node through Interface-I. If the current node is the destination node of the original data flow, the message will be dropped and no further unicast of the message will be sent.

3.4.4 Receiving a Data Packet

The data packets received by a switch agent can be either a data packet tagged with channel ID 1 (pass-through traffic through Interface-I) or a data packet tagged with

ID 0, which is a data packet originated from the current node. For the queue-length switching, if there is no registry entry for the destination of the data packet, the switch agent will forward it to RA-I and the packet will be tagged with channel ID 1. If the interface queue length exceeds the predefined threshold after receiving the packet, the switch agent will turn on the Interface-II if it is not currently on. The interface queue will be scanned and the destination with the maximum number of packets will be found. A registry entry will be added for that destination. For the delay-bound switching, if the current node is the destination node or an intermediate node other than the source node, the one-hop delay will be updated and a new end-to-end delay will be calculated and sent back to the previous hop if the difference between the current one-hop delay and the last updated one-hop delay exceeds a predefined range. If the end-to-end delay exceeds the upper-bound limit, the switch agent will turn on the Interface-II and forward the data packet to RA-II. Otherwise, the data packet is re-timestamped and forwarded to RA-I.

A broadcast/ringcast or a unicast wake-up message need to be sent to wake up downstream nodes to that destination. First look up the route cache to see if any route has been cached for the destination. If the answer is yes, construct a unicast wake-up packet and set its destination to be the next hop of the cached route and send the wake-up packet through Interface-I. If the answer is no, construct a broadcast wake-up packet and broadcast it through Interface-I.

The data packets received by a switch agent could also be tagged with ID 2. That indicates the Interface-II must be awake already. Otherwise, no packet tagged with

ID 2 can be received. The packet will be sent back down through the Interface-II to the next hop. A registry entry will be added for the destination of the packet, if it does not exist yet. The expiration time of the entry will be REG_TTL. If such an entry already exists it will be re-timestamped.

Some MAC protocols (like IEEE 802.11 and IEEE 802.15.4) and the routing agents (like AODV) support callback functions in case of transmission errors. Those callback functions can be leveraged to make sure the wake-up message can reach the downstream nodes. If any callback occurs on the unicast wake-up message due to collisions or poor link quality, an alternative path might be used or a broadcast/ringcast wake-up message will be sent out instead.

The pseudocode is listed in Procedures 3.1-3.5.

3.5 Performance Evaluation

3.5.1 Simulation Setup

We have carried out the simulations based on the *NS-2* [3] implementation of IEEE 802.15.4 PHY/MAC [4] (the interface with a lower data rate, shorter-range and lower power consumption) and IEEE 802.11 PHY/MAC [2] (the interface with a higher data rate and power consumption). The IEEE 802.11 MAC in *NS-2* only implements DCF function, without the complexity of association.

IEEE 802.15.4 standard has been developed to address the unique needs of low cost and low power of wireless sensor networks. IEEE 802.15.4 and IEEE 802.11

Algorithm 3.1 Pseudocode for receiving a packet

```
RECEIVE(pkt) {
//PT_RT: routing message from the routing agents
//PT_WK: wake-up message from the switch agent
//PT_DATA: data packets from the application
//RA-I: routing agent 1; RA-II: routing agent 2
nid ← current node id
ptype ← packet type of pkt
cid ← channel ID tagged in the header of pkt
dst ← destination address of pkt
prevhop ← previous hop address of pkt
ts ← time stamp when pkt is entering the previous hop
if ptype = PT_RT then
    if cid = 1 then
        hand pkt over to RA-I
    else if cid = 2 then
        hand pkt over to RA-II
    end if
else if ptype = PT_WK then
    call recvWakeup(pkt)
else
```

if the delay-bound switching **then**

call *updateDelay*(*pkt*)

if $cid = 0$ OR $cid = 1$ **then**

$ts \leftarrow$ CURRENT_TIME; $prevhop \leftarrow nid$

end if

end if

$found \leftarrow$ *registry_lookup*(*dst*)

if $found =$ TRUE **then**

setRegistryExpireTime(*dst*, REG_TTL)

call *sendThroughChannel2*(*pkt*)

else if $cid = 2$ **then**

cancel_idletimer( ) //cancel the idle_timer if it is active

registry_insert(*dst*) //insert a registry entry for *dst*

setRegistryExpireTime(*dst*, REG_TTL)

call *sendThroughChannel2*(*pkt*)

else if the queue-length switching **then**

call *queueLengthSwitch*(*pkt*)

else if the delay-bound switching **then**

call *delayBoundSwitch*(*pkt*)

end if

end if}

Algorithm 3.2 Pseudocode for receiving wakeup message

```
recvWakeup(pkt){
if pkt is a broadcast message then
    ttl ← the Time-To-Live of pkt
    if ttl < 0 then
        drop pkt
    else
        rq_src ← source where pkt is originated
        rq_bid ← broadcast id given by the source
        fresh ← bid_lookup(rq_src, rq_bid)
        if fresh = FALSE then
            drop pkt //obsolete wake-up message
        else
            bid_insert(rq_src, rq_bid) //store the latest broadcast id
            if registry is empty then
                resched_idletimer(IDLE_INTERVAL)
            end if
            wake up Interface-II
            ttl−− //decrement ttl by 1
            broadcast pkt
        end if
    end if
```

else if *pkt* is a unicast message **then**

rq_dst ← destination which the wake-up request is bound to

ip_dst ← destination address of *pkt*

if registry is empty **then**

resched_idletimer(IDLE_INTERVAL)

end if

if *rq_dst* = address of the receiving node **then**

wake up Interface-II and drop *pkt*

else if *ip_dst* = address of the receiving node **then**

wake up Interface-II

if there exists a cached route to *rq_dst* **then**

nexthop ← *cache_lookup*(*rq_dst*)

call *sendWakeup*(*rq_dst*, *nexthop*)

else

call *bcastWakeup*( )

end if

drop *pkt*

else

continue to propagate *pkt*

end if

end if}

Algorithm 3.3 Pseudocode for updating end-to-end delay

```
updateDelay(pkt) {
//DIFF: one hop delay variation triggering delay updates
nid ← current node id
src ← source address of pkt
dst ← destination address of pkt
cid ← channel ID tagged in the header of pkt
ts ← time stamp when pkt is entering the previous hop
delay_ent ← end-to-end delay entry to dst
if cid = 1 AND src ≠ nid then
    delay_ent = delay_lookup(dst)
    if delay_ent exists then
        one_hop_delay ← CURRENT_TIME - ts
        if | delay_ent.one_hop_delay - one_hop_delay | >= DIFF then
            delay_ent.one_hop_delay ← one_hop_delay
            delay ← delay_ent.delay + one_hop_delay
            send the new delay to the previous hop
        end if
    end if
end if
}
```

Algorithm 3.4 Pseudocode for queue-length switching

queueLengthSwitch(pkt){

//q: the interface queue between RA-I and MAC-I

tag the header of pkt with $cid = 1$

send pkt down through Interface-I

if $q.length >=$ THRESHOLD_HT **then**

if Interface-II is sleep **then**

wake up Interface-II

end if

if registry table is empty **then**

cancel_idletimer( ) //cancel the idle_timer if it is active

end if

$dst\_max \leftarrow findMax(q)$

$found \leftarrow registry\_lookup(dst\_max)$

if $found =$ TRUE **then**

call wakeupChannel2(dst_max)

registry_insert(dst_max) //add an entry for dst_max

setRegistryExpireTime(dst, REG_TTL)

end if

end if

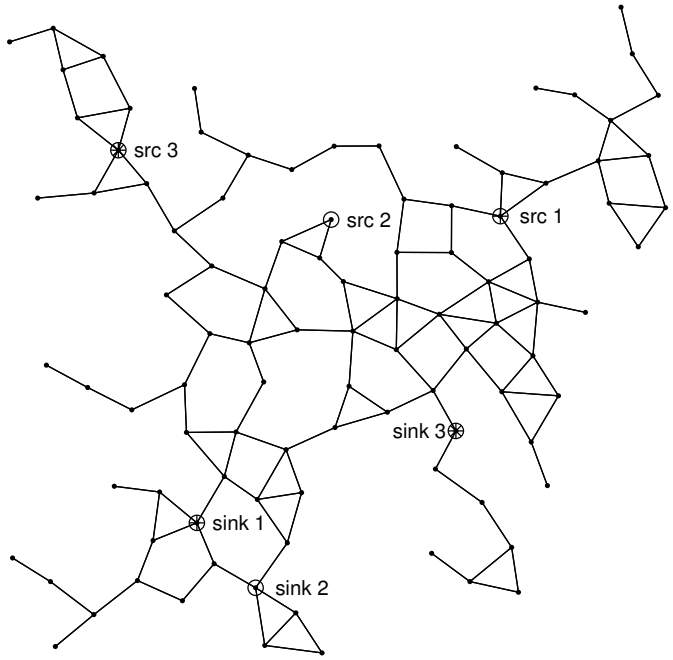
}

Algorithm 3.5 Pseudocode for delay-bound switching

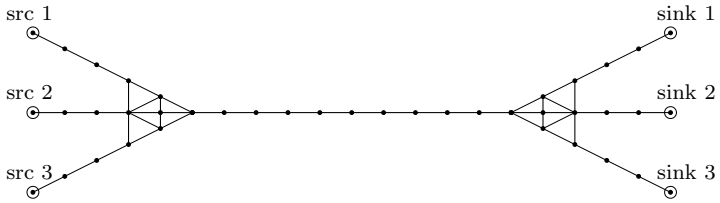
```
delayBoundSwitch(pkt){
delay_ent ← end-to-end delay entry to dst
tag the header of pkt with cid = 1
delay_ent ← delay_lookup(dst)
if delay_ent.delay >= DELAY_UPPER_BOUND then
    if Interface-II is sleep then
        call wake up Interface-II
    end if
    if registry table is empty then
        cancel_idletimer( ) //cancel the idle_timer if it is active
    end if
    call wakeupChannel2(dst)
    registry_insert(dst) //add a registry entry for dst
    setRegistryExpireTime(dst, REG_TTL)
else
    send pkt down through Interface-I
end if
}
```

both operate within the 2.4G ISM band. Interferences can occur when both types of devices coexist within a close region [50, 39]. However, two clear channels (25 and 26) exist outside the 802.11 spectrum and can be used as the primary 802.15.4 channels for interference-free deployment [21]. Given the above, in our simulations, we have assumed that the two interfaces operate in different channels with no interference between them.

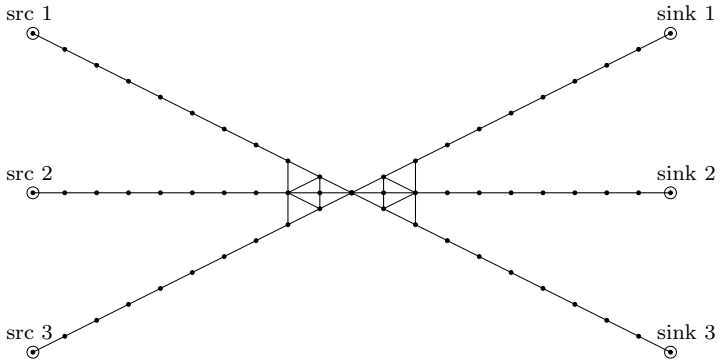
We have used on-demand routing protocol — AODV as the routing agents for both interfaces. We have simulated on a random topology as in Fig. 3.2(a). The random topology contains 100 nodes randomly generated with three source and destination pairs which share paths. All the nodes have at least one neighboring node within the transmission range of IEEE 802.15.4. We have also simulated on two other different topologies, one with a single joint path (Fig. 3.2(b)), the other with a single joint node (Fig. 3.2(c)). The simulation results turned out to have the same trends as the random topology. All the results reported here are for random topology unless otherwise specified. Bursty traffic was generated at each source node. The bursty traffic alternates between idle time period and burst time period. During the idle time period, no data was sent. During the burst time period, data were sent at a constant bit rate. The time span of the idle time period and the burst time period both follow a poisson distribution with a certain average. The data packet size is 70 bytes and the simulation time is set for 30 minutes. The THRESHOLD_HT is set to 32 packets. Some of the other parameters used in the simulations are listed in Table 3.1.



(a) Simulation topology with 100 nodes randomly generated.



(b) Simulation topology with a joint path.



(c) Simulation topology with a joint node.

Figure 3.2: Simulation topologies

Table 3.1: Parameters used in the simulations [1]

Parameter	IEEE 802.15.4	IEEE 802.11
transmit power	28.1 mW	660 mW
receiving power	62.1 mW	395 mW
idle power	1.4 mW	35 mW
data rate	250 kbps	2 Mbps
range	15 m	250 m

In the following discussions, the term "switch agent" refers to the scenario where sensor nodes have both IEEE 802.15.4 and IEEE 802.11 interfaces. The term "802.15.4 alone" refers to the scenario where sensor nodes with just the IEEE 802.15.4 interface alone are used. The term "802.11 alone" refers to the scenario where sensor nodes with just the IEEE 802.11 interface alone are used.

3.5.2 Performance Metrics

The following metrics have been used to evaluate the performance of the switch agent.

1. Average Goodput: the average number of bits (data packets only) received at a sink node within a unit of time;

2. Packet Delivery Ratio: the ratio of the number of data packets received over the number of data packets sent out;

3. Average End-to-End Delay: the average end-to-end delay between transmitting a data packet and receiving it at its destination;

4. Average Energy Consumption: the energy consumption of a single node on average;

5. Energy Efficiency: the ratio of the number of bits received at the sink nodes to the total energy consumed.

3.5.3 CBR Traffic

In this simulation, three pairs of flows with constant bit rate traffic start at different time. The packet size is 70 bytes. The simulation time is only 100 seconds, since it intends to show the switching moment only.

Fig. 3.3(a) shows that all the three types of interfaces have the same goodput when the interval is greater than 0.06 seconds. At that point, the traffic load exceeds the data rate limit of IEEE 802.15.4. Packets start dropping on the IEEE 802.15.4 interface. For the switch agent, the goodput remain close to that of the 802.11 alone, since the IEEE 802.11 interface has been turned on by the switch agent.

Fig. 3.3(b) shows the packet delivery ratio drops abruptly when the data rate limit of the IEEE 802.15.4 is reached, while the switch agent has as good delivery ratio as the 802.11 alone.

Fig. 3.3(c) shows the average end-to-end delay. The switch agent has the same end-to-end delay as IEEE 802.11. And both delays are lower than that of the 802.15.4 alone, since IEEE 802.15.4 has shorter transmission range and hence more

hops are involved. The spike at 0.06 seconds for IEEE 802.15.4 is due to the callback triggered by node failure.

Fig. 3.3(d) shows the average residual energy at each node. IEEE 802.11 consumes much more energy even when the traffic load is light, since its idle listening energy is much higher. When the traffic load exceeds the data rate limit of IEEE 802.15.4, its energy does not drop much since the IEEE 802.15.4 network is already saturated. However, the residual energy of the switch agent remain close to that of the IEEE 802.15.4, since the IEEE 802.11 interfaces were selectively waken up.

Fig. 3.3(e) shows the number of bits received on every unit of energy consumption. When traffic load is light, IEEE 802.15.4 and the switch agent have better energy efficiency than IEEE 802.11. When the traffic load is high, the switch agent remains highest efficiency among the three. It is even higher than that of the 802.11 alone since the IEEE 802.11 interfaces on the switch agents are selectively waken up.

3.5.4 Bursty Traffic with Varying Rates

Fig. 3.4 and Fig. 3.5 shows the simulation results for bursty traffic with varying rates on the random topology and the joint path topology, respectively. In this simulation, we have kept the average idle time and the average burst time to be fixed (both are 10 seconds), and investigated the performance under various traffic rates during the burst time period.

Fig. 3.4(a) shows that all the three scenarios yield the same average goodputs when the data rate is lower than 10Kbits/sec. For the data rate greater than

10Kbits/sec, the traffic load exceeds the data rate limit of IEEE 802.15.4. Packets start dropping for the 802.15.4 alone. For the switch agent, the goodput remains close to that of the 802.11 alone, since the IEEE 802.11 interface is turned on by the switch agent. Apparent goodput drop for both the 802.11 alone and the switch agent is observed on the joint path topology (Fig. 3.5(a)) when the data rate is greater than 40Kbits/sec. This is due to the reduced network capacity induced by increasing collisions along the joint path.

Fig. 3.4(b) shows the packet delivery ratio drops abruptly for the 802.15.4 alone when the data rate limit of IEEE 802.15.4 is reached, while the switch agent has as good delivery ratio as the 802.11 alone. Consistent with the goodput, the delivery ratio drops apparently on the joint path topology ((Fig. 3.5(b)) after 40Kbits/sec for both the 802.11 alone and the switch agent.

Fig. 3.4(c) shows a transition occurs on the end-to-end delay of the switch agent when the traffic load becomes heavier. It is the same as that of the 802.15.4 alone when the traffic load is light, while it gets close to that of the 802.11 alone when the traffic load increases. The end-to-end delay of the 802.15.4 alone increases dramatically when traffic load becomes heavy due to the increased collisions. The end-to-end delays of the 802.11 alone and the switch agent are also getting worse after 40Kbits/sec on the joint path topology (Fig. 3.5(c)).

Fig. 3.4(d) shows the average energy consumption at each node. The 802.11 alone consumes much more energy even when the traffic load is light, since its idle listening energy is much higher. When the traffic load exceeds the data rate limit of IEEE

802.15.4, its energy consumption does not increase much since the IEEE 802.15.4 network is already saturated. However, the energy consumption of the switch agent remains close to that of the 802.15.4 alone, since their IEEE 802.11 interfaces were selectively waken up.

Fig. 3.4(e) shows the number of bits received on every unit of energy consumption. When traffic load is light, the 802.15.4 alone and the switch agent have better energy efficiency than the 802.11 alone. The switch agent surpasses the 802.15.4 alone when the data rate reaches 25Kbits/sec and remains the highest efficiency among the three afterwards. That is because the IEEE 802.11 interfaces on the switch agent are selectively waken up.

3.5.5 Bursty Traffic with Varying Burst Time

Fig. 3.6 shows the simulation results for bursty traffic with varying burst time periods. In this simulation, we have kept the average idle time to be fixed (10 seconds), and investigated the performance under various average bursty time. The rate during burst time period is 10Kbits/sec.

Fig. 3.6(a) shows an increase of goodput with longer period of burst time for all the three scenarios. The switch agent has a goodput very close to that of the 802.11 alone, which is also much better than that of the 802.15.4 alone.

Fig. 3.6(b) shows that the switch agent has a delivery ratio close to that of the 802.11 alone, which is also close to 1. The delivery ratio of the 802.15.4 alone

decreases with the increase of the burst time period, because the longer burst time could cause more collisions.

Fig. 3.6(c) shows the average end-to-end delay. The 802.15.4 alone has the largest delay and the 802.11 alone has the smallest delay. The switch agent falls in between. When the burst time period increases, the delay of the switch agent is getting closer and closer to that of the 802.11 alone, since more traffic is being sent through the higher-bandwidth radio.

Fig. 3.6(d) shows the average energy consumption at each node. The switch agent consumes much less energy than the 802.11 alone, since only the nodes involved in data communication will stay awake. It is a little more than what the 802.15.4 alone consumes.

Fig. 3.6(e) shows the energy efficiency defined by the number of bits received on every unit of energy consumption. Since the switch agent has a goodput very close to the 802.11 alone but with a lot less energy consumption, it yields a much better energy efficiency compared to the 802.11 alone without compromising the throughput and end-to-end delay. The energy efficiency of the switch agent will surpass that of the 802.15.4 alone when the traffic load gets heavier.

3.5.6 Delay Bound Switching for Bursty Traffic with Varying Rates

Simulations have been carried out on all the three topologies for the delay-bound switching. Fig. 3.7 shows the simulation results for bursty traffic with varying rates. In this simulation, we have kept both the average idle time and the average burst

time to be 10 seconds and set the upper-bound limit of the end-to-end delay to be 1 second. Simulation results on all the three topologies exhibit the same trend.

Fig. 3.7(a) shows that all the three scenarios yield the same average goodputs when the data rate is lower than 5Kbits/sec. For data rate in between 10Kbits/sec and 20Kbits/sec, the average goodputs of both the 802.15.4 alone and the switch agent drops since the traffic load exceeds the data rate limit of IEEE 802.15.4. The switch agent drops even more than the 802.15.4 alone. This is introduced by the extra delay packets in the switch agent. Starting from 25Kbits/sec, the delay bound switching occurred at the switch agent, so its average goodput jumps close to the 802.11 alone.

Fig. 3.7(b) shows the packet delivery ratio drops abruptly for both the 802.15.4 alone and the switch agent after the data rate limit of IEEE 802.15.4 is reached. The switch agent drops even more than the 802.15.4 alone before the delay bound switching occurs at 25Kbits/sec. Again, this is due to the extra delay packets in the switch agent.

Fig. 3.7(c) shows that the switch agent gives a little higher end-to-end delay than the 802.15.4 alone before 25Kbit/sec due to the extra delay packets. Delay bound switching occurs starting from 25Kbits/sec and the end-to-end delay for the switch agent is getting close to that of the 802.11 alone. Even though the average end-to-end delay for the 802.15.4 alone is still below 0.2 seconds at the rate of 25Kbits/sec, the instantaneous end-to-end delay start exceeding the 1 second delay upper bound at some moment, which triggers the switching.

Fig. 3.7(d) shows the average energy consumption at each node. The extra delay packets in the switch agent do not introduce much more energy consumption than the 802.15.4 before the switching occurs. After switching occurs at the rate of 25Kbits/sec, the switch agent consumes a modestly more energy than the 802.15.4, which, however, is still far below the energy consumption of the 802.11 alone.

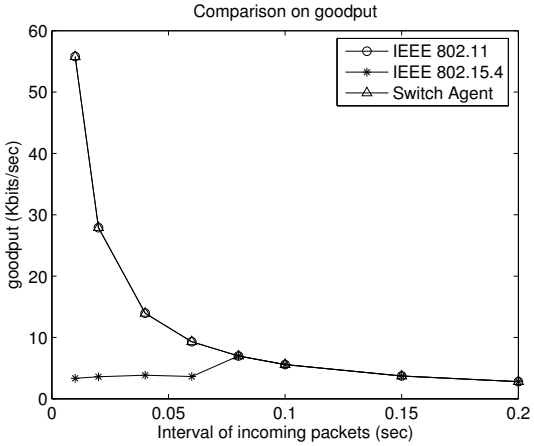
Fig. 3.7(e) shows the number of bits received on every unit of energy consumption. For both the random and the joint node topologies, the switch agent has higher energy efficiency than the 802.15.4 alone when the traffic load becomes heavy. That is not the case for the joint path topology (Fig. 3.7(f)), where its network capacity is relatively lower so that the traffic rate exceeds its network capacity. However, for all the three topologies, the switch agent shows higher energy efficiency than the 802.11 alone.

3.6 Summary

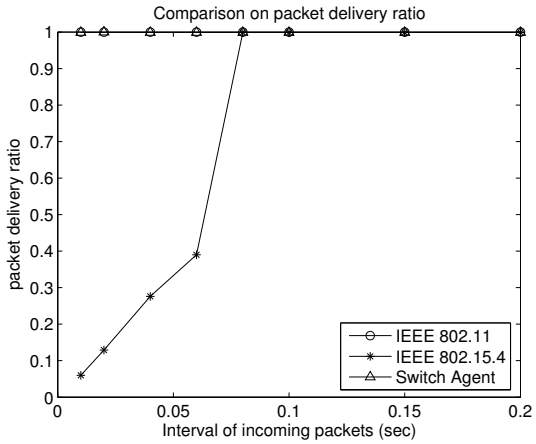
This chapter presents a switch agent at the routing layer, sitting on top of dual routing agents. The switch agent monitors the traffic flow or the end-to-end delay and switches on the high-bandwidth interface whenever the traffic rate becomes high or the end-to-end delay exceeds an upper bound. To save energy for using the high-bandwidth interface, the switch agent caches the routes established previously so that a unicast wake-up message can be sent out to selectively wake up the high-bandwidth interface at the downstream nodes. The switch agent also keeps a registry

for flows which already have the high-bandwidth interfaces awake so that no further wake-up message transmissions are incurred for subsequent requests.

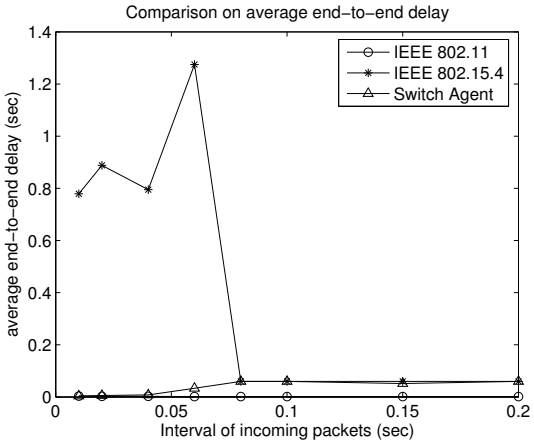
The simulations shows that the switch agent satisfies applications' demands on throughput and end-to-end delay without incurring much energy wastage, compared to the high-power radio alone.



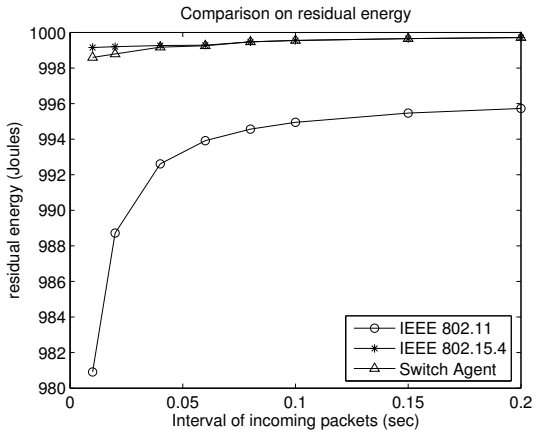
(a) Comparison on goodput



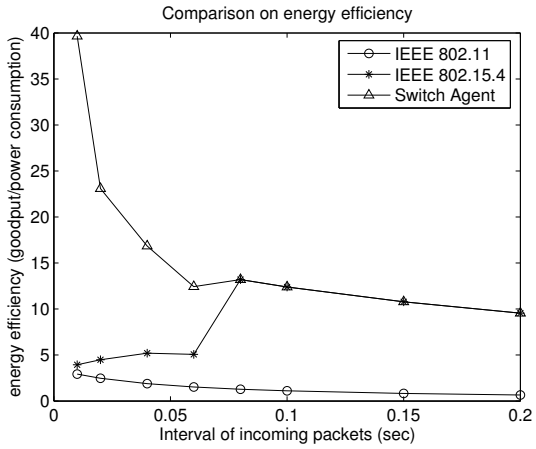
(b) Comparison on packet delivery ratio



(c) Comparison on average end-to-end delay.

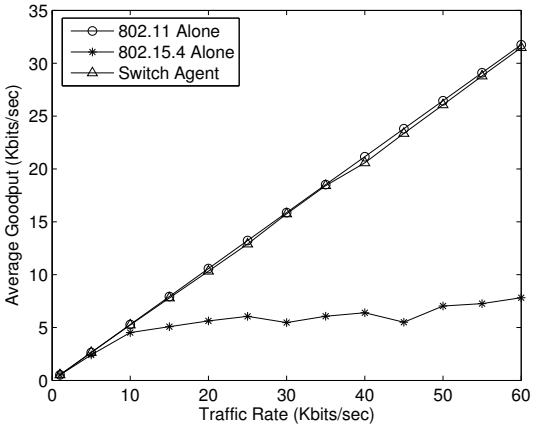


(d) Comparison on residual energy

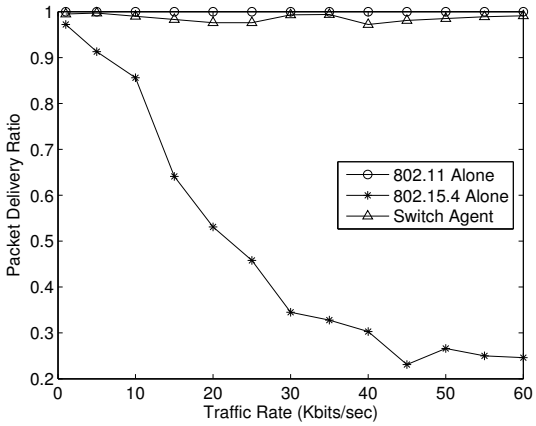


(e) Comparison on energy efficiency.

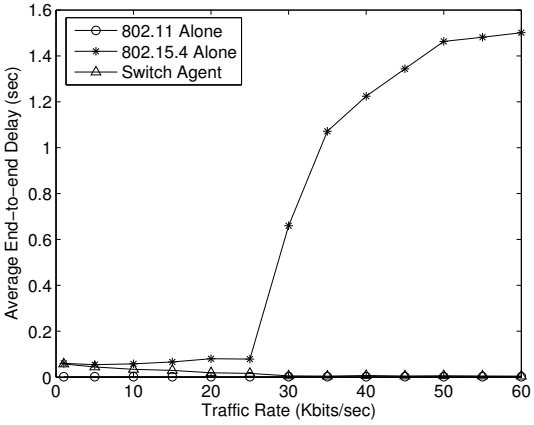
Figure 3.3: Comparisons among 802.11 alone, 802.15.4 alone and the switch agent on the random topology under different CBR traffic loads (for queue-length switching)



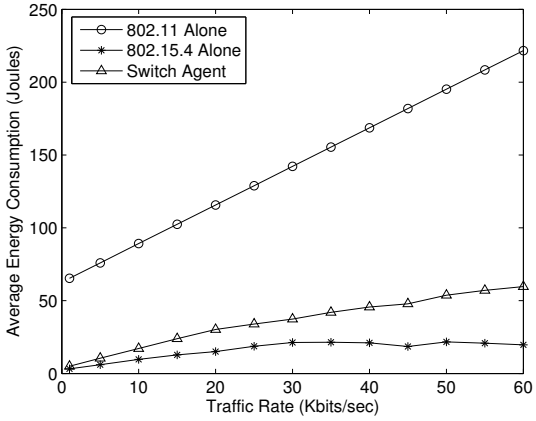
(a) Comparison on goodput



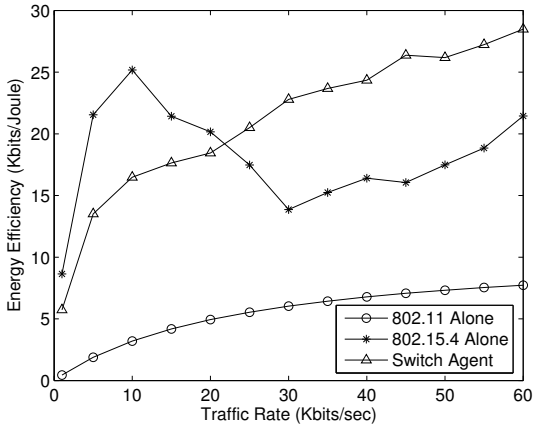
(b) Comparison on packet delivery ratio



(c) Comparison on average end-to-end delay

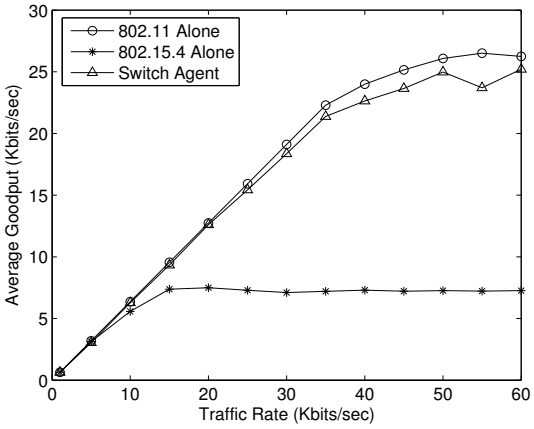


(d) Comparison on energy consumption

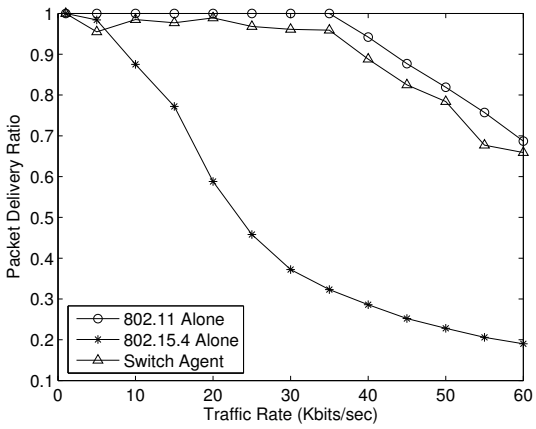


(e) Comparison on energy efficiency.

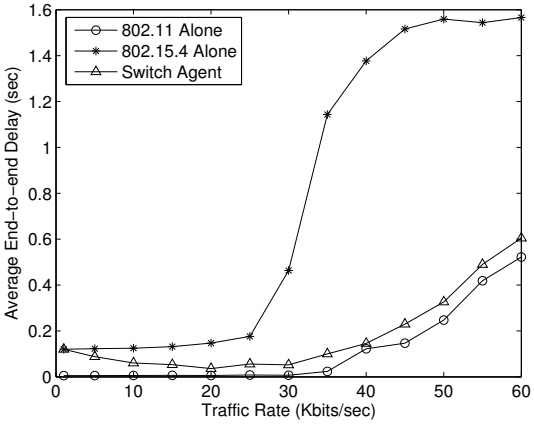
Figure 3.4: Comparisons among 802.11 alone, 802.15.4 alone and the switch agent on the random topology under different bursty traffic loads (for queue-length switching)



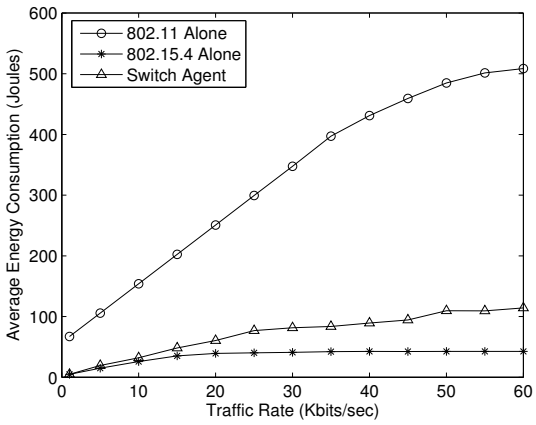
(a) Comparison on goodput



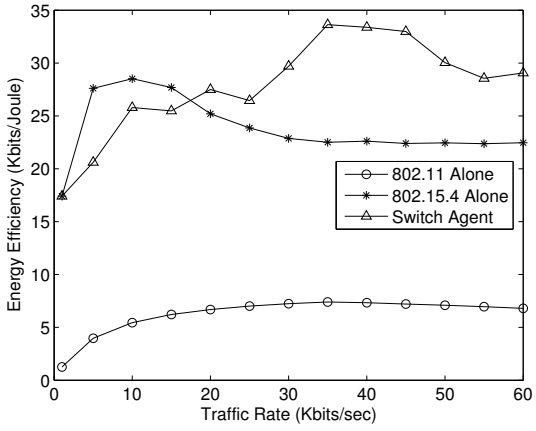
(b) Comparison on packet delivery ratio



(c) Comparison on average end-to-end delay

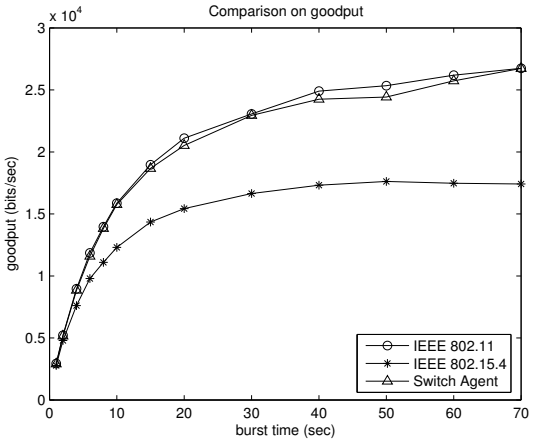


(d) Comparison on energy consumption

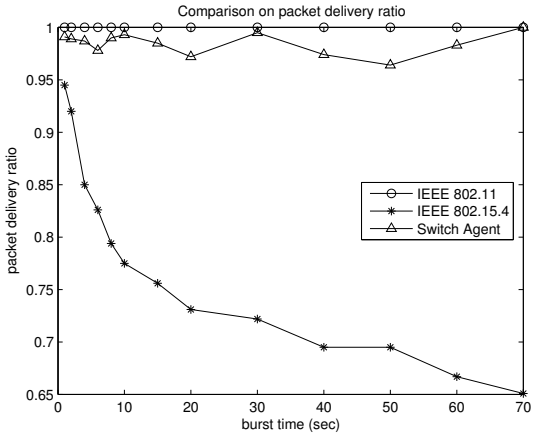


(e) Comparison on energy efficiency.

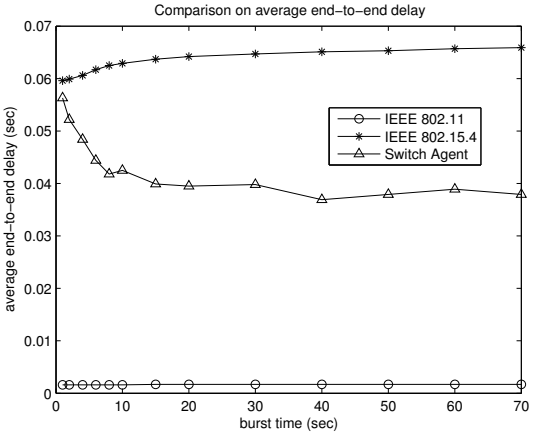
Figure 3.5: Comparisons among 802.11 alone, 802.15.4 alone and the switch agent on the joint path topology under different bursty traffic loads (for queue-length switching)



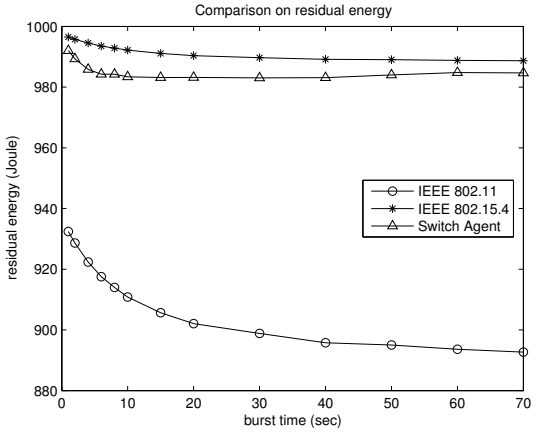
(a) Comparison on goodput



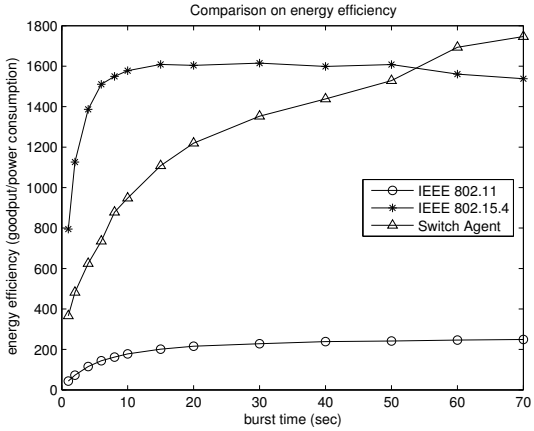
(b) Comparison on packet delivery ratio



(c) Comparison on average end-to-end delay.

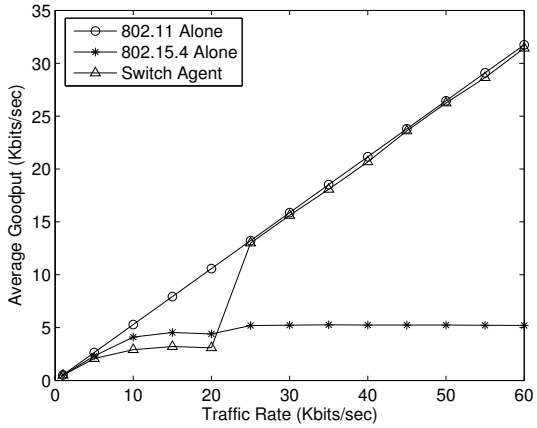


(d) Comparison on energy consumption

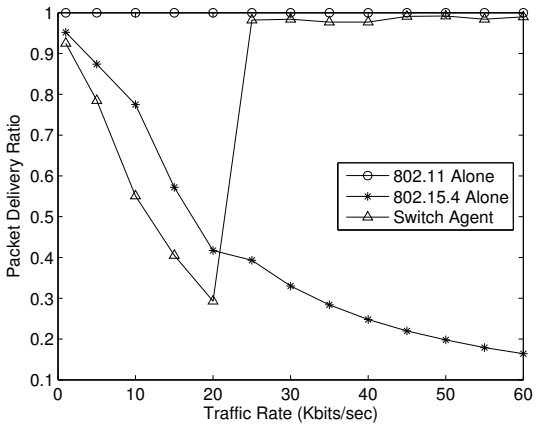


(e) Comparison on energy efficiency.

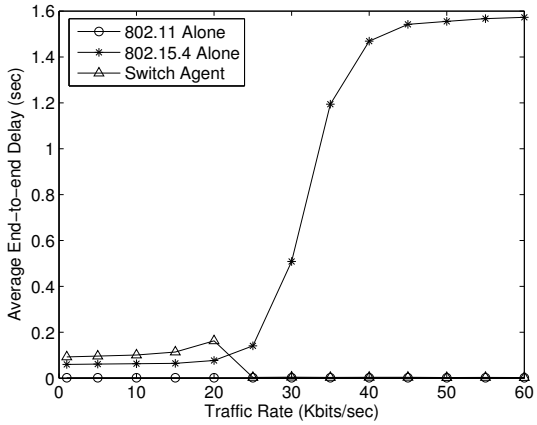
Figure 3.6: Comparisons among 802.11 alone, 802.15.4 alone and the switch agent on the random topology with different burst time (for queue-length switching)



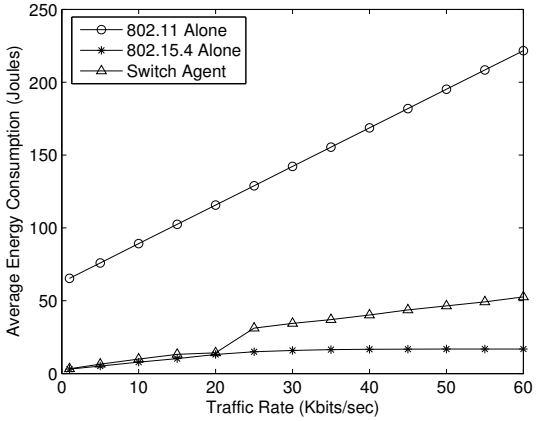
(a) Comparison on goodput



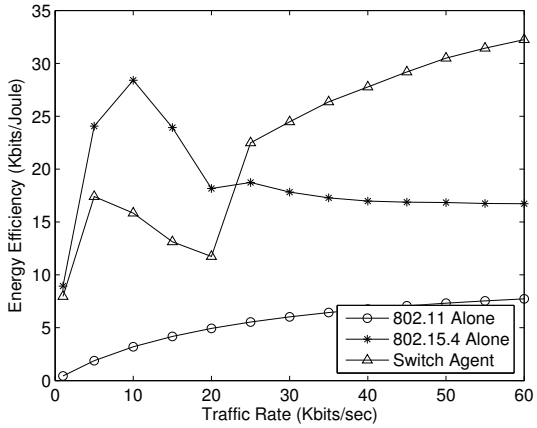
(b) Comparison on packet delivery ratio



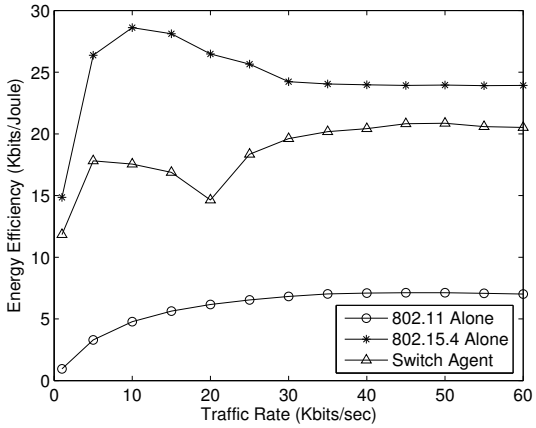
(c) Comparison on average end-to-end delay.



(d) Comparison on energy consumption



(e) Comparison on energy efficiency (random topology)



(f) Comparison on energy efficiency (joint-path topology)

Figure 3.7: Comparisons among 802.11 alone, 802.15.4 alone and the switch agent on the random topology under different bursty traffic loads (for delay-bound switching)

Chapter 4

Modeling and Performance Analysis of DMAC for WSNs

4.1 Introduction

Energy saving is one of the key challenges in wireless sensor networks due to infeasible battery replacement in most situations. Many MAC protocols have been proposed to save the unnecessary energy consumption due to idle listening by putting sensor nodes into sleep. SMAC [71], TMAC [56], BMAC [45] and XMAC [8] are some examples of the energy saving MAC protocols. DMAC [32] is another energy saving MAC protocol specifically designed for low-rate data gathering in wireless sensor networks. It employs *staggered* sleep-awake schedules to enable continuous data forwarding along a data gathering tree rooted at the sink node. Contention is reduced because the active periods are now separated. Additional active periods can be added to a node with little overhead through the *more data* flag when a node has multiple packets to send. A *data prediction* mechanism is used to add additional receiving and sending slots in order for other child nodes to send data in a timely manner. *More-To-Send* packets are used to avoid collision caused by interference between nodes on different branches of the data gathering tree.

Most of the work on energy saving MAC protocols for wireless sensor networks had taken the pragmatic and experimental approaches, with no analytical model presented to provide insights on how their protocols perform. Simulations are good for complex systems where the analytical solution is infeasible. However, simulation is quite time consuming and its results only demonstrate a few scenarios or instances and may not be enough to draw general conclusions. In this chapter, we present a generalized model for DMAC and analyze its performance under both CBR traffic and stochastic traffic following a Poisson process. The stochastic traffic scenario is modeled as a discrete time Markov chain and the analytical results are obtained using numerical methods. The consistency between the analytical results and the simulation results shows that the analytical approach can be used as a complementary tool for performance analysis on DMAC. The preliminary results of this work had been published in [75].

4.2 Related Work

Only a few recent work attempted to model and analyze some of the MAC protocols. Yang and Heinzelman had proposed Markov models for both SMAC [68] and XMAC [69]. In [68], two Markov models were proposed to evaluate the throughput of SMAC with and without retransmissions. The models were validated through simulations with varying number of sensor nodes, queue capacity, packet arrival rate and contention window. Similar approach is used to evaluate the throughput of

XMAC in [69]. Both analytical models match simulation results within a 5% range. Even though the lack of analytical work on energy saving MAC protocols for wireless sensor network, quite a lot of Markov models were proposed to analyze the IEEE 802.11 [6, 13, 19] and the IEEE 802.15.4 [20, 18, 46, 38, 42]. Bianchi [6] proposed a Markov model to analyze the saturation throughput of the IEEE 802.11 distributed coordination function, which inspired most of the analytical work coming after it. To the best of our knowledge, there is no analytical model proposed for evaluating the performance of DMAC.

4.3 A Generalized Model

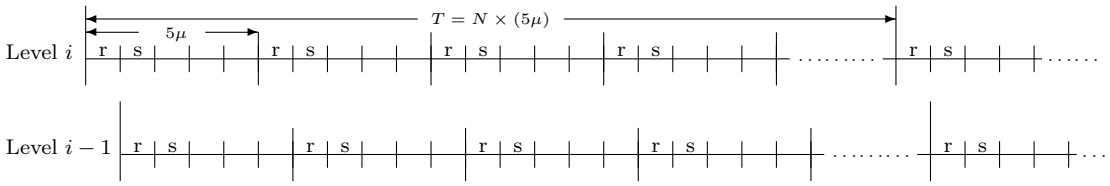


Figure 4.1: Time slots in DMAC

As shown in Fig. 4.1, DMAC is a time-slotted MAC protocol in which time is divided into small time slots. All the time slots have equal length and is long enough to send out one data packet along with control packets. Let μ be the length of such a time slot. In DMAC, a node will only send one data packet every 5 time slots in order to avoid collisions, so every 5 consecutive time slots can be virtually grouped into an *active period*. The first time slot in an active period is a receiving slot and labeled by r in Fig. 4.1, and the second slot in an active period is a sending slot

and labeled by s in Fig. 4.1. The other 3 time slots are all sleeping slots since the node knows the downstream nodes along a multi-hop path will forward the data packet in the next 3 time slots. Additional receiving/sending slots will be held in the subsequent active periods if (1) a *more data* flag is set in the current data packet because the sender has more data to send or (2) a sender has lost channel contention but overheard an ACK packet from its parent or (3) a node on a different branch is sending an explicit MTS packet.

A cycle, denoted by T, contains N number of active periods. That is, $T = N \times (5\mu)$. In the DMAC paper, $N = 4$. The receiving/sending time slots of a node at level $i-1$ is staggered with respect to level i by shifting one time slot to the right. Assume E_r is the energy consumed in the receiving time slot and E_s is the energy consumed in the sending time slot. Also assume that the energy consumed in the sending time slot is 0 if there is no data to be sent.

4.4 Analysis on DMAC

In this section, we analyze the average delay and average energy consumption at a single source node under both CBR and stochastic traffic following a Poisson process. For the sake of simplicity, we ignore the random delay introduced by contention and assume all the sending slots consume the same amount of energy in presence of data and same for all the receiving slots, however sending slots and receiving slots may consume different amount of energy.

4.4.1 CBR Traffic

CBR traffic can be regarded as a deterministic process. Let T_i be the time moment at which the ith packet arrives at a node, where $i = 0, 1, 2, ...$, and Δt denote the time interval between two consecutive data packets. Then we have

$$T_i = T_0 + i \times \Delta t \tag{4.1}$$

For the sake of simplicity, let's assume $T_0 = 0$ and $\Delta t = M\mu$ where $M = 1, 2, ...$ That is, $T_i = iM\mu$.

4.4.1.1 Case 1: $M \geq 5N$

This case can be regarded as $M = 5Nc+\delta$ with $c = 1, 2, 3, ...$ and $\delta = 0, 1, 2, ..., 5N-1$. In this case, the traffic load is so light that only the first receiving and sending slots in a T cycle are likely to be waken up to receive or send packets.

$$\begin{aligned} T_i &= i\mu(5Nc + \delta) \\ &= (ic + \left\lfloor \frac{i\delta}{5N} \right\rfloor)5N\mu + [i\delta - \left\lfloor \frac{i\delta}{5N} \right\rfloor 5N]\mu \\ &= (ic + \left\lfloor \frac{i\delta}{5N} \right\rfloor)5N\mu + \Delta_i \end{aligned} \tag{4.2}$$

where $\lfloor X \rfloor$ is the floor integer of a real number X and

$$\Delta_i = (i\delta - \left\lfloor \frac{i\delta}{5N} \right\rfloor 5N)\mu \tag{4.3}$$

Let D_i denote the delay of the ith packet at the node, then

$$D_i = \begin{cases} \mu - \Delta_i, & \text{if } 0 \leq \Delta_i \leq \mu \\ 5N\mu + (\mu - \Delta_i), & \text{if } \mu < \Delta_i < 5N\mu \end{cases} \tag{4.4}$$

Let $\tilde{D}$ denote the average delay over a large number of data packets. If $\delta = 0$, it is obvious that $\tilde{D} = \mu$. For $\delta \neq 0$, let $\frac{\delta}{5N} = \frac{m}{n}$ where m,n are positive integers and coprime. It is trivial to show that Δ_i is periodic with a period n since n is the smallest integer such that $\Delta_{i+n} = \Delta_i$. To calculate $\tilde{D}$, it is sufficient to calculate the average delay of all the n number of packets in a single period instead. If we can figure out how many packets out of the n number of packets fall into $0 \leq \Delta_i \leq \mu$ and how many fall into $\mu < \Delta_i < 5N\mu$, then we can calculate $\tilde{D}$ easily.

As Δ_i is a multiple of μ, $0 \leq \Delta_i \leq \mu$ is equivalent to $\Delta_i = 0$ or $\Delta_i = \mu$. That is,

$$\frac{i\delta}{5N} - \left\lfloor \frac{i\delta}{5N} \right\rfloor = 0 \tag{4.5}$$

or

$$\frac{i\delta}{5N} - \left\lfloor \frac{i\delta}{5N} \right\rfloor = \frac{1}{5N} \tag{4.6}$$

From Eqn. (4.5), we know that $\frac{i\delta}{5N} = \frac{im}{n}$ must be an integer and hence $i = Bn$ where $B = 0, 1, 2, ...$ It also means that within a period n, there is exactly one such i satisfying Eqn. (4.5). From Eqn. (4.6), we have

$$\begin{aligned} \frac{i\delta}{5N} &= C + \frac{1}{5N} \\ i &= \frac{5CN+1}{\delta}, \end{aligned} \tag{4.7}$$

where C is such an integer that $\frac{5CN+1}{\delta} = \lfloor \frac{5CN+1}{\delta} \rfloor$. Let A be the number of such C's also satisfying $i < n$, which can be computed numerically. Therefore, totally $(A+1)$ out of n number of packets fall into $0 \leq \Delta_i \leq \mu$ and the other $(n - A - 1)$ fall into $\mu < \Delta_i < 5N\mu$. Now the overall average delay $\tilde{D}$ at a source node can be expressed as

$$\tilde{D} = \frac{1}{n}\left[\sum_{i=0}^{n-1}((5N+1)\mu - \Delta_i) - 5N(A+1)\mu\right] \tag{4.8}$$

$$= \frac{\mu}{n}\left[n(5N+1) - 5N(A+1) - 5N\sum_{i=0}^{n-1}\left(\frac{im}{n} - \left\lfloor\frac{im}{n}\right\rfloor\right)\right]$$

$$= \frac{\mu}{n}\left[5N(n-A-1) + n - \frac{5Nm(n-1)}{2} + 5N\sum_{i=0}^{n-1}\left\lfloor\frac{im}{n}\right\rfloor\right] \tag{4.9}$$

The summation term in Eqn. (4.8) gives the total delay by assuming all the n number of packets fall into $\mu < \Delta_i < 5N\mu$, which overestimates the total delay by $5N(A+1)\mu$ as $(A+1)$ out of n number of packets fall into $0 \leq \Delta_i \leq \mu$. The second term deducts

the overestimated delay from the summation term. Applying the following formula for positive integers m and n which are co-prime,

$$\sum_{i=1}^{n-1} \left\lfloor \frac{im}{n} \right\rfloor = \frac{1}{2}(m-1)(n-1), \tag{4.10}$$

we have

$$\begin{aligned} \tilde{D} &= \frac{\mu}{n}\left[5N(n-A-1)+n-\frac{5Nm(n-1)}{2}+\frac{5N(m-1)(n-1)}{2}\right] \\ &= \frac{\mu}{n}\left[5N\left(\frac{n-1}{2}-A\right)+n\right] \end{aligned} \tag{4.11}$$

According to Eqn. (4.2), the arrival time of the nth packet will be $T_n = (nc+m)T$. That means there are $(nc+m)$ number of T cycles upon receiving the nth packet. The source node will be waken up at the first and only the first receiving time slot of every T cycle. Therefore, there are totally $(nc+m)$ awake receiving slots. Since only one packet can be sent in a cycle $T = 5N\mu$, only n number of sending time slots are really awake to send the n number of packets. The average energy consumed within a T cycle will be:

$$\begin{aligned} \tilde{E} &= \frac{1}{nc+m}\left[(nc+m)E_r + nE_s\right] \\ &= E_r + \frac{n}{nc+m}E_s \end{aligned} \tag{4.12}$$

4.4.1.2 Case 2: $5 <= M < 5N$

Let i_k and $i_{k'}$ denote the $i_k th$ and $i_{k'} th$ packets starting from 0. They are also the first packets in the kth and $k'th$ T cycles to be buffered and delayed to their next T cycles, respectively. That is, i_k and $i_{k'}$ will be the smallest integer satisfying the following condition:

$$i_k M\mu - [5Nk'\mu + \mu + 5(i_k - 1 - i_{k'})\mu] > 0, \tag{4.13}$$

wherein $i_k M\mu$ is the arrival time of the $i_k th$ packet, $i_k - i_{k'}$ is the number of packets buffered and sent out before the $i_k th$ packet. This condition basically states that the arrival time of the $i_k th$ packet must come after the beginning of the last sending time slot.

Eqn. (4.13) can be reduced to

$$i_k > \frac{5Nk' - 5i_{k'} - 4}{M - 5} \tag{4.14}$$

Since i_k is the smallest integer satisfying the above inequality, it can be expressed as

$$i_k = \left\lceil \frac{5Nk' - 5i_{k'} - 4}{M - 5} \right\rceil \tag{4.15}$$

It is obvious that the initial condition $i_1 = 1$. Let K be the first T cycle such that $i_K = 5NK/M$. If such a K exists, then the process follows a periodic pattern. All

the T cycles thereafter will repeat the first K number of T cycles. Such K can be numerically computed through the following iterations.

$$\begin{aligned}
\textit{initial condition:} \quad & k' = 1 \textit{ and } i_1 = 1 \\
\textit{iteration:} \quad & i_k = \left\lceil \frac{5Nk' - 5i_{k'} - 4}{M-5} \right\rceil \\
& k = k' + \left\lceil \frac{i_k M - 5Nk'}{5N} \right\rceil \\
& k' = k \\
\textit{stop condition:} \quad & i_k = 5Nk/M \textit{ or } k \textit{ is very large}
\end{aligned} \tag{4.16}$$

Let S be the set of k's obtained through the iterations, then the average delay at a node can be expressed as

$$\begin{aligned}
\tilde{D} = \frac{M\mu}{5NK} \sum_{k \in S} \Bigg[& \sum_{j=1}^{i_k - i_{k'}} (5(j-1)+1) + \sum_{j=i_k}^{\lfloor \frac{5Nk}{M} \rfloor} (5Nk - jM) \\
& - \sum_{j=\lceil \frac{5Nk'}{M} \rceil}^{i_k - 1} (jM - 5Nk') \Bigg].
\end{aligned} \tag{4.17}$$

$\tilde{D}$ can be calculated as i_k can be computed through the iterations.

The average energy consumed within a T cycle can be expressed as

$$\begin{aligned}
\tilde{E} &= \frac{1}{K} \left[\frac{5NK}{M} (E_r + E_s) + KE_r \right] \\
&= \frac{5N}{M} (E_r + E_s) + E_r.
\end{aligned} \tag{4.18}$$

$\frac{5NK}{M}$ number of packets are sent out in K number of T cycles, each packet consumes $E_r + E_s$ amount of energy. An extra receiving time slot in every T cycle is used before going to sleep.

4.4.1.3 Case 3: $M < 5$

This case represents a saturated and unstable state. The arrival time of the ith packet is $T_i = iM\mu$ and the sending time of the ith packet is $(i \times 5\mu + \mu)$. Therefore the average delay upon the sending of the kth packet is

$$\tilde{D} = \frac{1}{k}\sum_{i=0}^{k}[i(5-M)+1]\mu. \tag{4.19}$$

The average energy consumption within a T cycle is $\tilde{E} = N(E_r + E_s)$.

4.4.2 Stochastic Traffic

Now let us assume the packet arrivals follow a Poisson process $\{N(t) : t \geq 0\}$ with a rate parameter λ, where $N(t)$ is the number of packets that have arrived up to time t. Then the probability of k number of packets arriving in time interval $(t, t + \Delta t]$ is given by a Poisson distribution:

$$P[N(t+\Delta t) - N(t) = k] = \frac{e^{-\lambda\Delta t}(\lambda\Delta t)^k}{k!} \tag{4.20}$$

where $k = 0, 1, 2....$ Poisson process is memoryless, which means that the number of packets arriving after time t is independent of the number of packets arriving before time t.

X_{ij} denotes the state of a sensor node at the end of every T cycle, where $i = 0, 1, 2, ..., N$ and $j = 0, 1, 2, ...$ are two discrete valued random variables. Random variable i denotes that the first i number of active periods in the T cycle are used for receiving and sending. Random variable j denotes that there are j number of packets buffered at the end of the T cycle. Then the random variables $\{X_{ij}\}$ forms a Discrete Time Markov Chain since the next state only depends on the current state and not on the past. Let $P_{ij,mn}$ denote the state transition probability from state X_{ij} to X_{mn}. Let $D_{ij,mn}$, $E_{ij,mn}$ be the average delay and the energy consumption incurred at the state X_{mn}, respectively. Note that $D_{ij,mn}$, $E_{ij,mn}$ only take into account the effects of the packets arrived or buffered at the current T cycle and exclude the effects of the packets buffered at the previous T cycle. A few cases need be considered here.

4.4.2.1 Case 1: $0 \leq m < N$ and $m < j$

It is obvious that in this case the state transition probability $P_{ij,mn} = 0$. If $j < N$, then all the j packets buffered at the end of the previous T cycle are supposed to be sent out at the first j number of active periods of the current T cycle. If $j \geq N$, then the first N number of packets will be sent out in the current T cycle. Therefore, $m \geq j$ or $m = N$ is the necessary condition for having a non-zero state transition

probability $P_{ij,mn}$. All these states in this case should be excluded in the calculation of the average delay or the average energy consumption.

4.4.2.2 Case 2: $0 \leq j < m < N$

The state transition probability can be expressed as:

$$\begin{aligned} P_{ij,mn} &= e^{-\lambda\mu(5(m-1)+1)} \frac{[\lambda\mu(5(m-1)+1)]^{(m-j)}}{(m-j)!} \times I_{j,m} \\ &\quad \times e^{-\lambda\mu(5(N-m+1)-1)} \frac{[\lambda\mu(5(N-m+1)-1)]^n}{n!} \\ &= e^{-\lambda\mu(5N)} \frac{[\lambda\mu(5(m-1)+1)]^{(m-j)}}{(m-j)!} \times I_{j,m} \\ &\quad \times \frac{[\lambda\mu(5(N-m+1)-1)]^n}{n!} \end{aligned} \tag{4.21}$$

where $I_{j,m}$ is an integral depending on j and m:

$$\begin{aligned} I_{j,m} &= \frac{(m-j)!}{(5(m-1)+1)^{(m-j)}} \int_0^{5(j-1)+1} \int_{\tau_1}^{5j+1} \int_{\tau_2}^{5(j+1)+1} \\ &\quad ... \int_{\tau_{m-j-2}}^{5(m-3)+1} \int_{\tau_{m-j-1}}^{5(m-2)+1} d\tau_{m-j} d\tau_{m-j-1} ... d\tau_3 d\tau_2 d\tau_1 \end{aligned} \tag{4.22}$$

for $0 < \tau_1 < \tau_2 < \tau_3 < ... < \tau_{m-j-1} < \tau_{m-j} < 5(m-2)+1$.

In this case, in addition to the j number of packets from the previous T cycle, there must be another $(m-j)$ number of packets arriving during the first $5(m-1)+1$ time slots. The probability for that to occur is given by the terms before the $I_{j,m}$ term in Eqn. (4.21). However, this does not guarantee that all those $(m-j)$ packets will be sent out as supposed to be. There are further restrictions on the arrival time

of the $(m-j)$ packets in order for them to be sent. The probability of having proper arrival timings among those $(m-j)$ packets is given by the term $I_{j,m}$, which we will explain in details shortly. Furthermore, there should be another n number of packets coming and being buffered during the rest of the T cycle, whose probability is given by the terms after the term $I_{j,m}$.

In order to understand the term $I_{j,m}$, first let's recall the order statistics property of a Poisson process. For a Poisson process $\{N(t) : t \geq 0\}$, suppose we are given that for a fixed t, $N(t) = n$. Let T_i be the arrival time of the ith event, where $i = 1, 2, ..., n$. Let $u_1, u_2, ..., u_n$ be $i.i.d$ random variables each having a uniform distribution over $(0, t)$ and $U_{(1)}, U_{(2)}, ..., U_{(n)}$ be their order statistics. Then the conditional joint probability density function of $(T_1, T_2, ..., T_n)$ given that $N(t) = n$ is the same as the joint probability density function of $(U_{(1)}, U_{(2)}, ..., U_{(n)})$. That is,

$$\begin{aligned} & f_{T_1,T_2,...,T_n}((\tau_1, \tau_2, ..., \tau_n)|N(t) = n) \\ = & \begin{cases} \frac{n!}{t^n}, & \text{for } 0 < \tau_1 < \tau_2 < ... < \tau_n < t \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{4.23}$$

Without loss of generality, we assume $\mu = 1$. According to the order statistics property in Eqn. (4.23), the conditional joint probability density function of the $(m-j)$ packets is $\frac{(m-j)!}{(5(m-1)+1)^{(m-j)}}$, for $0 < \tau_1 < \tau_2 < \tau_3 < ... < \tau_{m-j-1} < \tau_{m-j} < 5(m-1)+1$. Furthermore, in order for the first packet among the $(m-j)$ packets to be sent out, it must arrive no later than the $(5(j-1)+1)th$ time slot. That is,

$0 < \tau_1 \leq 5(j-1)+1$ must be satisfied so that a *more data* flag will piggyback the jth data packet being sent out. As a result, the next receiving/sending time slots will stay awake to send more data. The second packet must come after τ_1 but no later than the $(5j+1)th$ time slot, i.e., $\tau_1 < \tau_2 \leq 5j+1$, and so forth. Integrating the conditional joint probability density function over all those ranges will give us the conditional joint probability for all the $(m-j)$ packets to be sent out.

The average delay $D_{ij,mn}$ can be expressed as

$$\begin{aligned} D_{ij,mn} &= e^{-\lambda\mu(5N)} \frac{[\lambda\mu(5(m-1)+1)]^{(m-j)}}{(m-j)!} \frac{\mu}{(m-j+n)} \\ &\times \left\{ A_{j,m} + I_{j,m} B_{m,N} + I_{j,m} \sum_{k=1}^{n} (5(k-1)+1) \right\} \\ &\times \frac{[\lambda\mu(5(N-m+1)-1)]^n}{n!} \end{aligned} \tag{4.24}$$

where $A_{j,m}$ is an integral depending on j and m:

$$\begin{aligned} A_{j,m} &= \frac{(m-j)!}{(5(m-1)+1)^{(m-j)}} \int_0^{5(j-1)+1} \int_{\tau_1}^{5j+1} \int_{\tau_2}^{5(j+1)+1} \\ &\ldots \int_{\tau_{m-j-2}}^{5(m-3)+1} \int_{\tau_{m-j-1}}^{5(m-2)+1} \sum_{k=1}^{m-j} (5(j+k-1)+1-\tau_k) \\ &\cdot d\tau_{m-j} d\tau_{m-j-1} \ldots d\tau_2 d\tau_1 \end{aligned} \tag{4.25}$$

for $0 < \tau_1 < \tau_2 < ... < \tau_{m-j-1} < \tau_{m-j} < (5(m-2)+1)$, and $B_{m,N}$ is an integral depending on m and N:

$$\begin{aligned} B_{m,N} &= \frac{n!}{(5(N-m+1)-1)^n} \int_{5(m-1)+1}^{5N} \int_{\tau_1}^{5N} \cdots \int_{\tau_{n-2}}^{5N} \\ &\quad \cdot \int_{\tau_{n-1}}^{5N} \sum_{k=1}^{n} (5N - \tau_k) d\tau_n d\tau_{n-1} ... d\tau_2 d\tau_1 \end{aligned} \tag{4.26}$$

for $5(m-1)+1 < \tau_1 < \tau_2 < ... < \tau_{n-1} < \tau_n < 5N$.

Inside the curly bracket of Eqn. (4.24), the $A_{j,m}$ term is associated with the delays of the $(m-j)$ number of packets received and to be sent in the current T cycle. The $B_{m,N}$ term is related to the sleep delays of the n number of buffered packets up to the end of the current T cycle. The last term is associated with the delays of the n number of buffered packets with respect to the beginning of the next T cycle. The $B_{m,N}$ term and the last term together give the delays of the n number of buffered packets.

4.4.2.3 Case 3: $0 \leq j = m < N$

The state transition probability can be expressed as:

$$\begin{aligned} P_{ij,mn} &= e^{-\lambda\mu(5(j-1)+1)} e^{-\lambda\mu(5(N-j+1)-1)} \times \frac{[\lambda\mu(5(N-j+1)-1)]^n}{n!} \\ &= e^{-\lambda\mu(5N)} \frac{[\lambda\mu(5(N-j+1)-1)]^n}{n!} \end{aligned} \tag{4.27}$$

A node must stay awake at the receiving and sending time slots of the first $5(j-1)+1$ number of time slots in the current T cycle in order to send all the j number of packets buffered during the previous T cycle. To end up at a state X_{mn} with $m = j$, there must be no packet arriving by the end of the first $(5(j-1)+1)$ time slots. The probability for that to happen is $e^{-\lambda\mu(5(j-1)+1)}$ according to the Poisson distribution in the Eqn. (4.20), wherein $k = 0$ and $\Delta t = (5(j-1)+1)\mu$. Also to end up at a state X_{mn}, there must be another n number of packets arriving during the rest of the T cycle after the first $(5(j-1)+1)$ time slots. The probability for that to happen is again given by the Poisson distribution in the Eqn. (4.20), wherein $k = n$ and $\Delta t = [5(N-j+1)-1]\mu$.

The delay $D_{ij,mn}$ in this case can be expressed as:

$$\begin{aligned} D_{ij,mn} &= e^{-\lambda\mu(5N)} \frac{[\lambda\mu(5(N-j+1)-1)]^n}{n!} \frac{\mu}{n} \\ &\times \left\{ C_{j,N} + \sum_{k=1}^{n} (5(k-1)+1) \right\} \end{aligned} \tag{4.28}$$

where $C_{j,N}$ is an integral depending on j and N:

$$\begin{aligned} C_{j,N} &= \frac{n!}{(5(N-j+1)-1)^n} \int_{5(j-1)+1}^{5N} \int_{\tau_1}^{5N} \cdots \int_{\tau_{n-2}}^{5N} \\ &\cdot \int_{\tau_{n-1}}^{5N} \sum_{k=1}^{n} (5N-\tau_k) d\tau_n d\tau_{n-1} ... d\tau_2 d\tau_1 \end{aligned} \tag{4.29}$$

for $5(j-1)+1 < \tau_1 < \tau_2 < ... < \tau_{n-1} < \tau_n < 5N$.

Inside the curly bracket of Eqn. (4.28), the $C_{j,N}$ term is associated with the sleep delays of the n number of buffered packets up to the end of the current T cycle. The last term is related to the delays of the n number of buffered packets with respect to the beginning of the next T cycle.

4.4.2.4 Case 4: $0 \leq j < m = N$

The state transition probability can be expressed as:

$$P_{ij,mn} = e^{-\lambda\mu(5N)} \frac{[\lambda\mu(5N)]^{(N-j+n)}}{(N-j+n)!} \times I_{j,N} \tag{4.30}$$

where $I_{j,N}$ is an integral depending on j and N:

$$\begin{aligned} I_{j,N} &= \frac{(N-j+n)!}{(5N)^{(N-j+n)}} \int_0^{5(j-1)+1} \int_{\tau_1}^{5j+1} \int_{\tau_2}^{5(j+1)+1} \\ &\ldots \int_{\tau_{N-j-2}}^{5(N-3)+1} \int_{\tau_{N-j-1}}^{5(N-2)+1} \int_{\tau_{N-j}}^{5N} \int_{\tau_{N-j+1}}^{5N} \\ &\ldots \int_{\tau_{N-j+n-1}}^{5N} d\tau_{N-j+n} \ldots d\tau_{N-j+1} d\tau_{N-j} \ldots d\tau_3 d\tau_2 d\tau_1 \end{aligned} \tag{4.31}$$

for $0 < \tau_1 < \tau_2 < \ldots < \tau_{N-j-1} < \tau_{N-j} < \ldots < \tau_{N-j+n-1} < 5N$.

In this case, in addition to the j number of packets from the previous T cycle, there must be another $(N-j+n)$ number of packets arriving during the current T cycle. The first $(N-j)$ ones among the $(N-j+n)$ number of packets will be sent out resulting in a state X_{Nn}. The probability to have $(N-j+n)$ number of packets coming during the current T cycle is given by the terms before the $I_{j,N}$ term

in Eqn. (4.30). Again, this does not guarantee that its first $(N-j)$ packets will be sent out as supposed to be. There is further restriction on the arrival time of the $(N-j+n)$ packets in order for the first $(N-j)$ packets to be sent. The probability of having proper arrival timings among those $(N-j+n)$ packets is given by the term $I_{j,N}$.

Without loss of generality, we again assume $\mu = 1$. According to the order statistics property in Eqn. (4.23), the conditional joint probability density function of the $(N-j+n)$ packets is $\frac{(N-j+n)!}{(5N)^{(N-j+n)}}$, for $0 < \tau_1 < \tau_2 < ... < \tau_{N-j-1} < \tau_{N-j} < ... < \tau_{N-j+n-1} < 5N$. Furthermore, in order for the first packet among the $(N-j+n)$ packets to be sent out, it must arrive no later than the $(5(j-1)+1)th$ time slot. That is, $0 < \tau_1 \leq 5(j-1)+1$ must be satisfied. The second packet must come after τ_1 but no later than the $(5j+1)th$ time slot, i.e., $\tau_1 < \tau_2 \leq 5j+1$. It repeats until $\tau_{N-j-1} < \tau_{N-j} \leq 5(N-2)+1$. After that, $\tau_{k-1} < \tau_k < 5N$, for $k = N-j+1, N-j+2, ..., N-j+n$. Integrating the conditional joint probability density function over all those ranges will give us the conditional joint probability for all the $(N-j+n)$ packets to be sent out.

The delay $D_{ij,mn}$ can be expressed as

$$\begin{aligned} D_{ij,mn} &= e^{-\lambda\mu(5N)} \frac{[\lambda\mu(5N)]^{(N-j+n)}}{(N-j+n)!} \frac{\mu}{(N-j+n)} \\ &\times \left\{ F_{j,m} + I_{j,N} \sum_{k=1}^{n} (5(k-1)+1) \right\} \end{aligned} \tag{4.32}$$

where $F_{j,m}$ is an integral depending on j and m:

$$\begin{aligned} F_{j,m} &= \frac{(N-j+n)!}{(5N)^{(N-j+n)}} \int_0^{5(j-1)+1} \int_{\tau_1}^{5j+1} \int_{\tau_2}^{5(j+1)+1} \\ &\ldots \int_{\tau_{N-j-2}}^{5(N-3)+1} \int_{\tau_{N-j-1}}^{5(N-2)+1} \int_{\tau_{N-j}}^{5N} \int_{\tau_{N-j+1}}^{5N} \ldots \int_{\tau_{N-j+n-1}}^{5N} \\ &\cdot \left(\sum_{k=1}^{N-j} (5(j+k-1)+1-\tau_k) + \sum_{k=N-j+1}^{N-j+n} (5N-\tau_k)\right) \\ &\cdot d\tau_{N-j+n} \ldots d\tau_{N-j+1} d\tau_{N-j} \ldots d\tau_3 d\tau_2 d\tau_1 \end{aligned} \tag{4.33}$$

for $0 < \tau_1 < \tau_2 < ... < \tau_{N-j} < ... < \tau_{N-j+n} < 5N$.

The first summation term of $F_{j,m}$ in Eqn. (4.33) is associated with the delays of the $(N-j)$ number of packets received and to be sent in the current T period. The second summation term of $F_{j,m}$ is related to the sleep delays of the n number of buffered packets up to the end of the current T cycle. Inside the curly bracket of Eqn. (4.32), the last term is associated with the delays of the n number of buffered packets with respect to the beginning of the next T cycle.

4.4.2.5 Case 5: $N = m \leq j \leq N+n$

The state transition probability can be expressed as:

$$P_{ij,mn} = e^{-\lambda\mu(5N)} \frac{[\lambda\mu(5N)]^{(N-j+n)}}{(N-j+n)!} \tag{4.34}$$

In this case, more than N number of packets are buffered by the end of the previous T cycle, but only N out of them can be sent out. The other $(j-N)$ number of

packets stay in the buffer. To end up with a state at X_{Nn}, there must be another $(n-j+N)$ number of packets coming during the current T cycle. The probability for that to occur is given by the Eqn. (4.20), wherein $k = n-j+N$ and $\Delta t = 5N\mu$.

The delay $D_{ij,mn}$ can be expressed as

$$\begin{aligned} D_{ij,mn} &= e^{-\lambda\mu(5N)} \frac{[\lambda\mu(5N)]^{(N-j+n)}}{(N-j+n)!} \frac{\mu}{(N-j+n)} \\ &\times \left\{ G_{j,m} + \sum_{k=j-N+1}^{n} (5(k-1)+1) \right\} \end{aligned} \tag{4.35}$$

where $G_{j,m}$ is an integral depending on j and m:

$$\begin{aligned} G_{j,m} &= \frac{(N-j+n)!}{(5N)^{(N-j+n)}} \int_0^{5N} \int_{\tau_1}^{5N} \int_{\tau_2}^{5N} \cdots \int_{\tau_{N-j+n-1}}^{5N} \\ &\cdot \sum_{k=1}^{N-j+n} (5N-\tau_k) d\tau_{N-j+n} ... d\tau_3 d\tau_2 d\tau_1 \end{aligned} \tag{4.36}$$

for $0 < \tau_1 < \tau_2 < ... < \tau_{N-j+n} < 5N$.

Inside the curly bracket of Eqn. (4.35), The $G_{j,m}$ term is associated with the delays of the $(N-j+n)$ number of packets arrived at the current T cycle up to the end of the current T cycle. The second term is associated with the delays of the $(N-j+n)$ buffered with respect to the beginning of the next T cycle.

4.4.2.6 Case 6: $m = N, j >= 2N$

This case represents a saturated and unstable state. It occurs only when the traffic load exceeds the channel capacity. Since no equilibrium state in this case, Markov

model is no longer applicable. Therefore, we assume the traffic load is less than the channel capacity and hence ignore this case.

The state transition probabilities and the delay can be summarized as Eqn. (4.37) and Eqn. (4.38), respectively.

$$P_{ij,mn} = \begin{cases} 0 & \text{if } 0 \le m < N \text{ and } m < j \\ e^{-\lambda\mu(5N)} \frac{[\lambda\mu(5(m-1)+1))]^{(m-j)}}{(m-j)!} & \\ \times I_{j,m} \times \frac{[\lambda\mu(5(N-m+1)-1))]^n}{n!} & \text{if } 0 \le j < m < N \\ e^{-\lambda\mu(5N)} \frac{[\lambda\mu(5(N-j+1)-1))]^n}{n!} & \text{if } 0 \le j = m < N \\ e^{-\lambda\mu(5N)} \frac{[\lambda\mu(5N)]^{(N-j+n)}}{(N-j+n)!} \times I_{j,N} & \text{if } 0 \le j < m = N \\ e^{-\lambda\mu(5N)} \frac{[\lambda\mu(5N)]^{(N-j+n)}}{(N-j+n)!} & \text{if } N = m \le j \le N+n \end{cases} \tag{4.37}$$

$$D_{ij,mn} = \begin{cases} 0 & \text{if } 0 \le m < N \text{ and } m < j \\ e^{-\lambda\mu(5N)} \frac{[\lambda\mu(5(m-1)+1))]^{(m-j)}}{(m-j)!} & \\ \times \frac{[\lambda\mu(5(N-m+1)-1))]^n}{n!} \frac{\mu}{(m-j+n)} & \\ \times \{A_{j,m} + I_{j,m} B_{m,N} & \\ + I_{j,m} \sum_{k=1}^{n} (5(k-1)+1)\} & \text{if } 0 \le j < m < N \\ e^{-\lambda\mu(5N)} \frac{[\lambda\mu(5(N-j+1)-1))]^n}{n!} \frac{\mu}{n} & \\ \times \{C_{j,N} + \sum_{k=1}^{n} (5(k-1)+1)\} & \text{if } 0 \le j = m < N \\ e^{-\lambda\mu(5N)} \frac{[\lambda\mu(5N)]^{(N-j+n)}}{(N-j+n)!} \frac{\mu}{(N-j+n)} & \\ \times \{F_{j,m} + I_{j,N} \sum_{k=1}^{n} (5(k-1)+1)\} & \text{if } 0 \le j < m = N \\ e^{-\lambda\mu(5N)} \frac{[\lambda\mu(5N)]^{(N-j+n)}}{(N-j+n)!} \frac{\mu}{(N-j+n)} & \\ \times \left\{ G_{j,m} + \sum_{k=j-N+1}^{n} (5(k-1)+1) \right\} & \text{if } N = m \le j \le N+n \end{cases} \tag{4.38}$$

Let π_{ij} be the equilibrium probability that the system is in the state X_{ij}, then the global balance equations for the state X_{ij} can be expressed as

$$\pi_{ij} \sum_{m=0}^{N} \sum_{n=0}^{M} P_{ij,mn} = \sum_{m=0}^{N} \sum_{n=0}^{M} \pi_{mn} P_{mn,ij} \tag{4.39}$$

The sum of all the probabilities of leaving a state should always be 1, *i.e.*,

$$\sum_{m=0}^{N}\sum_{n=0}^{M} P_{ij,mn} = 1 \tag{4.40}$$

Therefore, Eqn. (4.39) can be reduced as:

$$\pi_{ij} = \sum_{m=0}^{N}\sum_{n=0}^{M} \pi_{mn} P_{mn,ij} = \sum_{n=0}^{M}\sum_{m=0}^{N} \pi_{mn} P_{mn,ij} \tag{4.41}$$

Since the state transition probability $P_{mn,ij}$ is independent of m, Eqn. (4.41) can be written as

$$\pi_{ij} = \sum_{n=0}^{M} \left(P_{0n,ij} \sum_{m=0}^{N} \pi_{mn} \right) \tag{4.42}$$

Let's define

$$\hat{\pi}_n = \sum_{m=0}^{N} \pi_{mn} \tag{4.43}$$

for $n = 0, 1, 2...$, then Eqn. (4.42) becomes

$$\pi_{ij} = \sum_{n=0}^{M} P_{0n,ij} \hat{\pi}_n \tag{4.44}$$

We further have

$$\hat{\pi}_j = \sum_{i=0}^{N} \pi_{ij} = \sum_{i=0}^{N}\sum_{n=0}^{M} P_{0n,ij} \hat{\pi}_n = \sum_{n=0}^{M} \left(\sum_{i=0}^{N} P_{0n,ij} \right) \hat{\pi}_n \tag{4.45}$$

In matrix notation, this becomes, with $\hat{\boldsymbol{\pi}}$ the column vector with elements $\hat{\pi}_n$,

$$\hat{\boldsymbol{\pi}} = \boldsymbol{Q}\hat{\boldsymbol{\pi}} \tag{4.46}$$

or

$$(\boldsymbol{I} - \boldsymbol{Q})\hat{\boldsymbol{\pi}} = \boldsymbol{0} \tag{4.47}$$

The determinant of $\boldsymbol{I} - \boldsymbol{Q}$ will be 0. To have a unique non-zero solution of $\hat{\boldsymbol{\pi}}$, replace one of the equation in Eqn. (4.47) with the normalization equation:

$$\sum_{n=0}^{M} \hat{\pi}_n = 1 \tag{4.48}$$

Now we have reduced a set of $(M+1) \times (M+1)$ equations into a set of $(M+1)$ equations instead. Solving Eqn. (4.47) together with Eqn. (4.48) is much easier than solving Eqn. (4.41). After we have $\hat{\boldsymbol{\pi}}$, we can obtain the probability π_{ij} that the system is in the state X_{ij} from Eqn. (4.44).

The average sleep delay incurred in the state X_{mn} can be expressed as

$$\tilde{D}_{mn} = \frac{\sum_{i=0}^{N} \sum_{j=0}^{M} \pi_{ij} D_{ij,mn}}{\pi_{mn}} \tag{4.49}$$

The overall average delay is

$$\tilde{D} = \frac{\sum_{m=0}^{N} \sum_{n=0}^{M} \pi_{mn} \tilde{D}_{mn}}{1 - \hat{\pi}_0} \tag{4.50}$$

Assume the sink node is at level 0, then the average end-to-end delay for a source node at level L will be

$$D_{end-end} = \tilde{D} + (L-1)\mu \tag{4.51}$$

The energy consumption $E_{ij,mn}$ can be expressed as

$$E_{ij,mn} = \begin{cases} 0, & \text{for } 0 \leq m < N, m < j \\ E_r, & \text{for } m = j = n = 0 \\ (m - j + n)(E_r + E_s), & \text{otherwise} \end{cases} \tag{4.52}$$

Similarly, the average energy consumed in the state X_{mn} can be expressed as

$$E_{mn} = \frac{\sum_{i=0}^{N} \sum_{j=0}^{M} \pi_{ij} P_{ij,mn} E_{ij,mn}}{\pi_{mn}} \tag{4.53}$$

The overall average energy consumption in a T cycle is

$$\tilde{E} = \sum_{m=0}^{N} \sum_{n=0}^{M} \pi_{mn} E_{mn} \tag{4.54}$$

4.5 Performance Evaluation

This section presents and compares the numerical and simulation results for both CBR and stochastic traffic.

4.5.1 Numerical Evaluation

Even though the integrals in Eqn. (4.37) and Eqn. (4.38) are not closed-form expressions, they all can be evaluated numerically with little efforts. For example, the $I_{j,m}$ can be computed through a small MATLAB function shown in the Appendix A.

Table 4.1: State transition probabilities (N=4)

	(0,0)	(0,1)	(0,2)	(0,3)	(0,4)
(0,0)	0.6792	0	0	0	0
(0,1)	0.2496	0	0	0	0
(0,2)	0.0459	0	0	0	0
(0,3)	0.0056	0	0	0	0
(0,4)	0.0005	0	0	0	0
(1,0)	0.0131	0.6792	0	0	0
(1,1)	0.0048	0.2496	0	0	0
(1,2)	0.0009	0.0459	0	0	0
(1,3)	0.0001	0.0056	0	0	0
(1,4)	0.0000	0.0005	0	0	0
(2,0)	0.0001	0.0131	0.6792	0	0
(2,1)	0.0000	0.0036	0.1839	0	0
(2,2)	0.0000	0.0005	0.0249	0	0
(2,3)	0.0000	0.0000	0.0022	0	0
(2,4)	0.0000	0.0000	0.0002	0	0
(3,0)	0.0000	0.0014	0.0788	0.6792	0
(3,1)	0.0000	0.0002	0.0137	0.1182	0
(3,2)	0.0000	0.0000	0.0012	0.0103	0
(3,3)	0.0000	0.0000	0.0001	0.0006	0
(3,4)	0.0000	0.0000	0.0000	0.0000	0
(4,0)	0.0000	0.0002	0.0122	0.1445	0.6792
(4,1)	0.0000	0.0001	0.0031	0.0405	0.2627
(4,2)	0.0000	0.0000	0.0004	0.0060	0.0508
(4,3)	0.0000	0.0000	0.0000	0.0006	0.0066
(4,4)	0.0000	0.0000	0.0000	0.0000	0.0006
Total	1.0000	1.0000	1.0000	1.0000	0.9999

All the other integrals in Eqn. (4.37) and Eqn. (4.38) can be evaluated numerically in similar ways. Table 4.1 lists the state transition probabilities for $N = 4$ as stated in the DMAC paper [32]. The packet arrival rate $\lambda = 2$ packets/second and the time slot $\mu = 0.00967$ second. Column j lists the state transition probabilities from the state X_{0j} to every other state, for $j = 0, 1, 2, 3, 4$. The summation of each column is equal or very close to 1 as expected. The state probabilities are listed in Table 4.2 and it is easy to verify that all the state probabilities add up to 1.

Table 4.2: Probabilities at state X_{ij} (N=4)

	0	1	2	3	4
0	0.4735	0.1740	0.0320	0.0039	0.0004
1	0.1798	0.0661	0.0121	0.0015	0.0001
2	0.0343	0.0093	0.0013	0.0001	0.0000
3	0.0077	0.0013	0.0001	0.0000	0.0000
4	0.0018	0.0005	0.0001	0.0000	0.0000
$\hat{\pi}_j$	0.6971	0.2512	0.0456	0.0055	0.0005

4.5.2 Simulation Setup

We have carried out the simulations on a line topology with 11 nodes using the *NS-2*. The data packet size is 70 bytes and the simulation time is set to 60 minutes. Table 4.3 shows some parameters used in both the DMAC paper [32] and in our simulations as well.

Table 4.3: Parameters used in the simulations

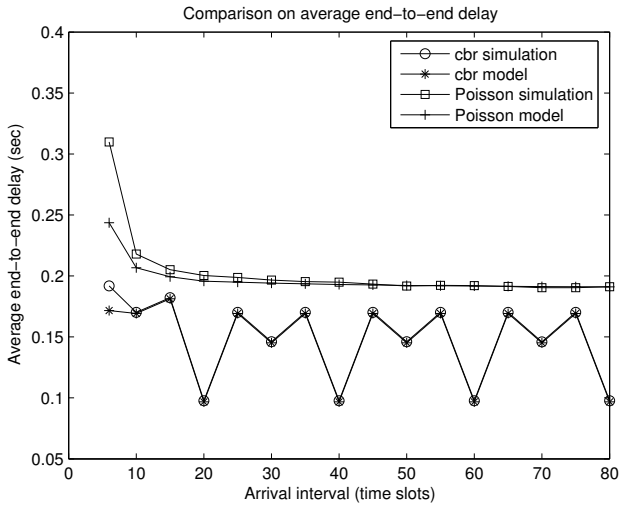
Parameter	IEEE 802.11
transmit power	660 mW
receiving power	395 mW
idle power	35 mW
data rate	2 Mbps
range	250 m

4.5.3 Simulation Results

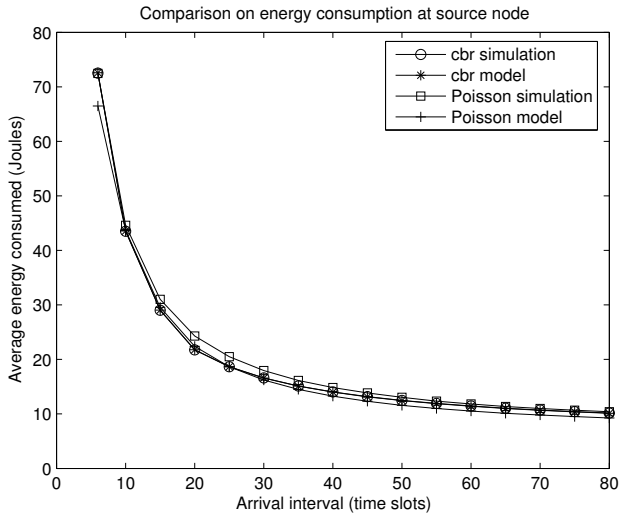
Fig. 4.2 compares the numerical and simulation results on average end-to-end delays and energy consumption with varying arrival intervals for both CBR and stochastic traffic.

Fig. 4.2(a) shows the numerical results accurately match the simulation results for CBR traffic with intervals equal or larger than 10μ. An interesting periodic and symmetric pattern is observed for CBR traffic for the selected time intervals within the same range. The numerical results for stochastic traffic are also close to the simulation results for large arrival intervals. As the interval is getting smaller, the stochastic model requires more states in order to have an accurate prediction, which explains the growing gap between the numerical and simulation results for stochastic traffic. Larger delays are observed for stochastic traffic at the selected arrival rates compared to CBR traffic, which indicates DMAC adapts better to CBR traffic than stochastic traffic at those arrival rates. The absence of the periodic pattern in

stochastic traffic reveals that stochastic traffic is less sensitive to the changes of intervals than CBR traffic is.



(a) Comparison on average end-to-end delay

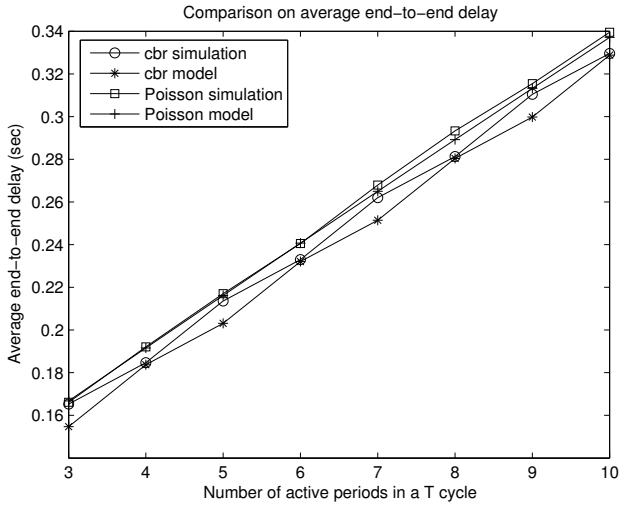


(b) Comparison on average energy consumption

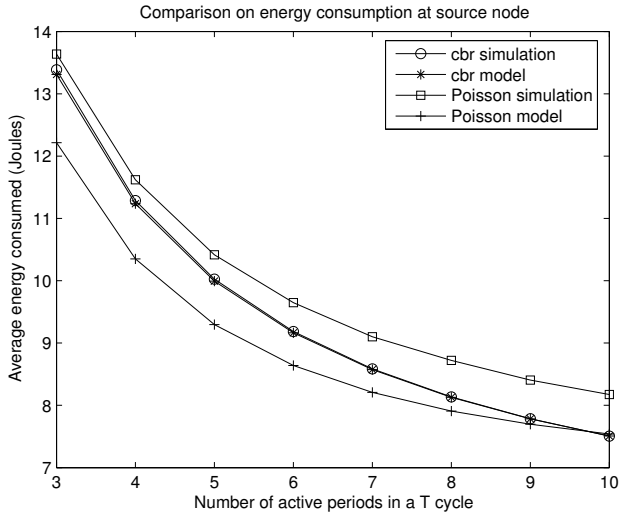
Figure 4.2: Comparisons under different traffic loads

In Fig. 4.2(b), the numerical results are also close to the simulation results when the arrival interval is large. The absence of oscillation in energy consumption for

CBR traffic indicates that the energy consumption is less sensitive to the changes of intervals than the delay is.



(a) Comparison on average end-to-end delay



(b) Comparison on average energy consumption

Figure 4.3: Comparisons with different number of active periods in a T cycle

Fig. 4.3 shows the effect of varying number of active periods in a T cycle on average end-to-end delays and energy consumption. The traffic interval is fixed to 0.6 seconds. Fig. 4.3(a) shows the numerical results of the delay match the simulation

results pretty well for both CBR and stochastic traffic. The delay increases as the number of active periods increases for both traffic conditions.

Fig. 4.3(b) shows the energy consumption decreases as the number of active periods in a T cycle increases for both traffic conditions. Even though stochastic traffic does not match its numerical results with its simulation results as well as CBR traffic does, the same tendency is still observed. From Fig. 4.3(a) and 4.3(b), we can also see the tradeoff between the delay and the energy consumption with the selection of different number of active periods in a T cycle.

4.6 Summary

In this chapter, we proposed generalized models of the DMAC protocol for both CBR traffic and stochastic traffic following a Poisson process. A discrete-time Markov chain is used to model the stochastic traffic scenario. The average delay and energy consumption at a source node are expressed and can be evaluated numerically. The close match between the numerical results and the simulation results validates the correctness of the models. The models provide insight on the adaptivity of DMAC under CBR and stochastic traffic conditions and indicate that DMAC only adapts varying traffic conditions to a limited extent due to the low channel utilization and DMAC adapts CBR traffic better than the stochastic traffic. The stochastic model can also be applied to scenarios with multiple source nodes.

Chapter 5

A Routing-Layer Sleep Scheme for Data Gathering in WSNs

5.1 Introduction

Data gathering is a typical operation in wireless sensor networks. As discussed in the previous chapter, DMAC [32] is a popular energy saving MAC protocol specifically designed for low-rate data gathering in wireless sensor networks. In this chapter, we present a sleep scheme at the routing layer instead of the MAC layer, which could possibly be incorporated with some of the routing-layer data gathering tree formation algorithms. By placing it at the routing layer, the sleep scheme can utilize the data gathering tree structures formed with existing algorithms. Furthermore, different existing MAC protocols can be used without modifications to suite wireless sensor networks. The key idea of the sleep scheme is to periodically wake up all the sensor nodes at the same time. Sleeping starts from the leaf nodes where no data to be sent and then shrinks inward towards the sink node along the data gathering tree. It turns out only the path involved in data communication will stay awake. This saves energy without compromising much on the throughput and the end-to-end delay.

5.2 Related Work

DMAC employs a staggered sleep-awake schedule to enable continuous data forwarding on the multi-hop path, which reduces the end-to-end delay introduced by sleep delay. Contention is reduced because the active periods are now separated. Additional active period can be added to a node with little overhead through the *more data* flag when a node has multiple packets to send. *Data prediction* scheme is used to add additional receiving and sending slots in order for other child nodes to send data in a timely manner. *More-To-Send* packet is used to avoid collision caused by the interference between nodes on different branches of the data gathering tree. DMAC is a very energy-efficient MAC protocol tweaked for data gathering when the data rate is very low.

At the routing layer, besides the two basic tree structures — the Shortest Path Tree (SPT) and the Minimum Spanning Tree (MST) [54], lots of more tree-based algorithms [33, 36, 30, 37] had been proposed for constructing the data gathering tree. Meghanathan [36] proposed a Connected Dominating Set (CDS) based data gathering algorithm, which prefers to include nodes with relatively high energy as the backbone nodes and nodes with relatively low energy as leaf nodes, yielding longer network life than that observed in classical cluster-based algorithms. Meghanathan [37] also proposed an energy-aware maximal leaf nodes data gathering algorithm, which minimizes the number of intermediate nodes. Chen *et al.* [14] proposed a dynamic and adjustable tree structure which has similar amount of children

among non-leaf nodes. The tree structure can be readjusted when the energy level of non-leaf nodes falls below a certain threshold. In contrast with plenty of work on data gathering tree constructions, very few work was presented on incorporating sleep schemes into those routing-layer algorithms. A routing layer sleep scheme was proposed in [67] on directed diffusion [22] and it shows that the 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 networks.

5.3 Protocol Details

In this section, we describe our routing-layer data gathering sleep scheme (DGSS) in details. We use DMAC as a baseline for comparisons and illustration.

5.3.1 Data Gathering Tree and Neighbor List

As mentioned previously, a data gathering tree could be a shortest path tree, a minimum spanning tree, a maximal leaf tree or a tree with a minimal connected dominating sets as its backbone, etc. For illustration purpose, we use the shortest path tree as a data gathering tree. Every node maintains a neighbor list which stores information about its neighbor nodes. The neighbor list is virtually the routing table.

We will adopt the same hierarchical level scheme as in [23]. The sink node has a level 0. Initially, the sink node will broadcast a *HELLO* message. The message

contains the level id and the sink node id. All the nodes at its immediate neighborhood will receive this message and will have a level 1. The sink node id will be added to the neighbor lists of all level 1 nodes as their parent and its corresponding "Active" flag is set to 1, which indicates the radio status. Then all the nodes at level 1 will broadcast a message with their own level id and node id after waiting for a random amount of time (to avoid collision). When the sink node receives such message, it will drop it. If any node at level 1 receives the message, no new level id will be assigned since it is already labeled with a level id. However, it will add the node id into its neighbor list and mark it as "Active". If any node not yet labeled with a level, it will be labeled with a level incremented by 1 and src node id will be added into its neighbor list as its next-hop to the sink and marked as "Active".

In general, let $N_{i,j}$ denote the node id of the jth node at level $i > 1$, and $\{N_{i-1}\}$, $\{N_i\}$, $\{N_{i+1}\}$ denote the set of nodes at level $i-1$, i, $i+1$ which can hear from the node $N_{i,j}$, respectively. That means $\{N_{i-1}\}\bigcup\{N_i\}\bigcup\{N_{i+1}\}$ contains all the neighbors of node $N_{i,j}$. When node $N_{i,j}$ broadcasts a message containing (N_{ij}, i), where $N_{i,j}$ is the node id and i is the level id, node $x \in\{N_{i-1}\}\bigcup\{N_i\}\bigcup\{N_{i+1}\}$ will add it into its neighbor list. The neighbor list at node $N_{i,j}$ will look like:

Table 5.1: Neighbor list at node $N_{i,j}$

Dest	Nexthop	Nexthop Level	Parent	Child	Active
0	$N_{i-1,x}$	i-1	1	0	1
0	$N_{i,x}$	i	0	0	1
0	$N_{i+1,x}$	i+1	0	1	1

Wherein the table 5.1, $N_{i-1,x} \in \{N_{i-1}\}$, $N_{i,x} \in \{N_i\}$, $N_{i+1,x} \in \{N_{i+1}\}$. The neighbor list is virtually a routing table, in which the parent node will be the next hop. Eventually all the active paths together will form a shortest path tree.

5.3.2 Active Leaf Nodes

A node can be identified as *leaf node* of the data gathering tree if the node's neighbor list does not contain any node with a higher level. This can be done locally and solely based on the neighbor list. A node $N_{i,j}$ is an *active leaf node* at time moment t if it is awake and it has no child node or all its child nodes are asleep. Active leaf node is the leaf node of a subtree at a certain time moment which consists of only active nodes. A sleep message always initiates at an active leaf node. If no traffic at an active leaf node for a certain period of time, a sleep request will be broadcasted at that node. As a comparison, DMAC behaves like a radar scans level-by-level to see if any data to send. It adapts to the traffic condition by scanning a little more frequently if there is more data to be sent. DGSS starts at the leaf nodes and scans radially inward towards the sink node.

5.3.3 Packet Queue

In traditional networks, the routing layer resolves the next-hop IP address of the incoming packets as soon as they are received at the routing layer. Then packets will be sent to logical link layer to resolve the MAC address of the next-hop through ARP. The packets are then sent down to the MAC layer and buffered at the interface

queue in between the logical link layer and the MAC layer, since the service rate of the MAC layer may be much slower than the rate of IP address resolution and ARP resolution. This mechanism works fine with the underlying assumption that the next hop is always available. However, for the duty-cycled wireless sensor networks, it is no longer true. The next hop may be in sleep for some time. In DGSS, we buffer the incoming packets in a queue before resolving the next-hop address. We make the rate of dequeuing to be in sync with the serving rate of the MAC layer through a MAC layer callback function. That is, the next packet will not be dequeued until the current packet has been sent out by the MAC layer. This way, we control the IP address resolution rate at the routing layer. If the next hop is in sleep, we don't have to keep routing packets to the sleeping next-hop. It can be rerouted to an awake next-hop instead.

The length of the packet queue will have the accumulative effects of both incoming traffic rate and MAC serving rate. It could be used as a good indication of current network condition.

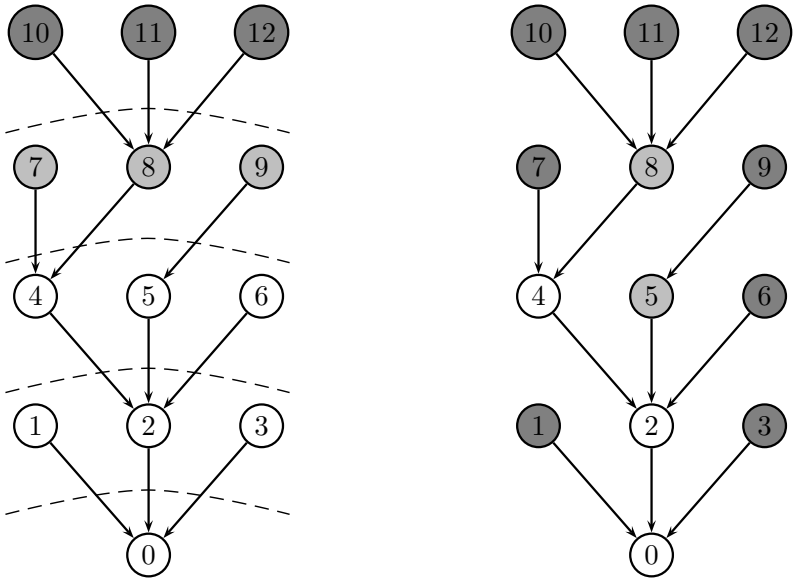
5.3.4 Sleep Scheme

In DGSS, all the nodes periodically wake up at the same time. This fits existing data gathering tree algorithms well as most of the algorithms requires tree reconstruction periodically. In DMAC, even though the wake-up time is staggered and not all the sensor nodes wake up at the same time, every node will wake up at least once in a

cycle. Therefore, waking up at the same time does not necessarily increase energy consumption compared to staggered wake-up.

In DGSS, the sleep starts from the leaf nodes and shrink inward towards the sink along the data gathering tree. If a leaf node does not have data in its buffer, it will broadcast a sleep message out to tell its neighbors that it will sleep till the next cycle and then goes to sleep immediately. There is a random delay before broadcasting the sleep message to reduce the chance of collision at the MAC layer. The randomness is proportional to the number of active nodes in its neighbor list. As illustrated in Fig. 5.1(b), the leaf nodes (in dark gray) are not necessarily at the same level. After the leaf nodes go to sleep, it appears as if they are pruned from the data gathering tree. All the active nodes still form a tree structure. Now node 8 and node 5 (in light gray color) become the active leaf nodes. The same process repeats until all nodes with no data go to sleep. As shown in Fig. 5.1(a), DMAC scans level by level, starting from the outmost level all the way towards to sink.

In DGSS, once a parent node receives a sleep message from its child node, it will update the entry corresponding to the child node at its neighbor list by marking the active flag to 0. After the parent node receives sleep messages from all its children, it will become an active leaf node. In case of no data to be sent, it will broadcast a sleep message. If there is no traffic along a path, a sleep message will quickly propagate towards the sink along the path. When neighboring nodes other than its parent node receive the sleep message, they will update their neighbor list and set the active flag of the sleeping node to be inactive. Every time a node wakes up, all



(a) Illustration of DMAC (b) Illustration of DGSS

Figure 5.1: Comparison between DMAC and DGSS

the entries in its neighbor list will be reset to be active again since all the nodes wake up at the same time.

5.3.5 Local Rerouting

Rerouting in DGSS only occurs locally. A different next-hop node will be selected to merge traffic together if the traffic is low. This will increase the channel utilization on the merged path so that idle listening energy waste could be reduced. Meanwhile the nodes along the old path can go to sleep and save even more energy.

In DGSS, local rerouting is triggered by receiving a sleep message from its child nodes. Let's say when node $N_{i,j}$ received a sleep message from one of his child nodes $N_{i+1,x} \in \{N_{i+1}\}$. If the number of packets in its DGSS queue is below a predefined

threshold, then node $N_{i,j}$ will broadcast a merge request to all its neighbors. The threshold is proportional to the number of child nodes. Only its child nodes in $\{N_{i+1}\}$ will handle the merge request. Once child nodes receive the merge request, they will search through their own neighbor lists and try to find their new active parents. If such parents are found, they will send a join request to their new parents. Their new parent will mark the join requester as its child and send back a join acknowledgement. Upon receiving the join acknowledgement, the join requester will mark the new parent active and then send back an acknowledgement to the merge request. The merge requester will then mark its child node to be inactive. After the merge requester receives acknowledgements from all its active child nodes, it will become an active leaf node. It will send out sleep message and go to sleep if there is no more packet buffered in its queue.

5.3.6 Position of the Sleep Scheme

Simple data gathering algorithms, like the Shortest Path Tree algorithm, can be implemented at the MAC layer instead of the routing layer because only single-hop broadcasts are needed to form such a simple data gathering tree. In that case, the DGSS can be implemented at the MAC layer too and thus becomes a new MAC protocol for single-sink data gathering. The neighbor list will be maintained at the MAC layer and leaf nodes can be identified at the MAC layer as well. Data packets will be sent down to the MAC layer directly without specifying the next hop. The default next hop will be the parent node of the current node in the data

gathering tree. We separate out the sleep scheme and place it at the routing layer so that it can fit into more sophisticated tree formation algorithms where single-hop broadcasts are not enough to form the data gathering tree. Furthermore, it can sit on top of different MAC protocols.

5.3.7 Further Comparison with DMAC

DMAC is better than DGSS in terms of energy saving no matter what tree structures are under consideration under light traffic condition. The reason is that in DMAC only nodes at the two adjacent levels involved in data communication will stay awake. In the extreme case of no traffic at all, every node in the data gathering tree needs stay awake for only one time slot in DMAC, while only the leaf nodes of the data gathering tree will stay awake for very short of time period in DGSS. The downstream nodes have to stay awake until they receive the sleep message from all the leaf nodes. To make DGSS close to DMAC in terms of energy saving, tree structures with maximal leaf nodes evenly distributed at different levels will help. DGSS has better performance than DMAC in terms of throughput and end-to-end delay under heavy traffic conditions. Take the same tree structure as shown in Fig. 5.1 and assume node 11 is the only source node with heavy traffic. In DGSS, only the nodes along the path $11 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 0$ will remain active all the time till all the data packets at node 11 are sent out. In DMAC, every node along the path $11 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 0$ will wake up every 5 time slots. Apparently, DGSS will yield higher throughput and smaller end-to-end delay than DMAC in this case.

Both DMAC and DGSS depend on some kind of tree formation algorithms. DMAC relies on its routing layer protocol to tell it what is its level in the data gathering tree and who is the next hop, while DGSS keeps the node id, level id and radio status of its neighbors in a table structure. Therefore, it needs a little bit more memory for the table structure than DMAC. DGSS identifies the (active) leaf nodes solely based on the table structure. The cpu time for finding the (active) leaf nodes is insignificant. And the sleep message in DGSS is a small size control packet. It only requires a little bit more memory and cpu time as well. In a word, DGSS incurs only a little extra resource usage in terms of memory and cpu compared to DMAC, but neither is significant.

5.4 Performance Evaluation

5.4.1 Simulation Setup

We have carried out the simulations in *NS-2*. The same topology shown in Fig. 5.1 has been used in the simulation with node 0 as the only sink node. Bursty traffic was generated at each source node. The bursty traffic alternates between idle time period and burst time period. During the idle time period, no data was sent. During the burst time period, data were sent at a constant bit rate. The time span of the idle time period and the burst time period both follow an exponential distribution with an average idle time of 5 seconds and an average burst time of 2 seconds. It appears that a random subset of the source nodes may be sending data at a given

time moment. All nodes wake up every 10 seconds. The data packet size is 70 bytes and the simulation time is set for 100 seconds. Some of the other parameters used in the simulations are listed in Table 5.2.

Table 5.2: Parameters used in the simulations

Parameter	IEEE 802.11
transmit power	660 mW
receiving power	395 mW
idle power	35 mW
data rate	2 Mbps
range	250 m

5.4.2 Performance Metrics

The following metrics have been used to evaluate the performance of DGSS.

1. Average Goodput: the average number of bits (data packets only) received at a sink node within a unit of time;

2. Packet Delivery Ratio: the ratio of the number of data packets received over the number of data packets sent out;

3. Average End-to-End Delay: the average end-to-end delay between transmitting a data packet and receiving it at its destination;

4. Average Energy Consumption: the energy consumption of a single node on average;

5. Energy Efficiency: the ratio of the number of bits received at the sink nodes to the total energy consumed.

5.4.3 Simulation Results

Fig. 5.2 compares the simulation results for DGSS, DGSS with rerouting, IEEE 802.11, DMAC and DMAC with More to Send (MTS).

Fig. 5.2(a) shows that the throughput of DMAC is saturated at very low traffic rate. This is because that DMAC is using staggered sleep schedules. DMAC scans through every level to see if there is any data to send or not after every fixed period of time. Only one node among any five sequential nodes along a path can send data at a certain moment. This reduces the channel utilization. DMAC is good only for very low rate communication. Throughput is not the target of DMAC. The design goal is to save energy by sacrificing the channel utilization. Both DGSS and DGSS with rerouting demonstrate the throughput is close to that of the IEEE 802.11. Fig. 5.2(b) shows that the delivery ratios of DGSS and DGSS with rerouting are close to that of the IEEE 802.11 and much better than DMAC at higher rates.

Fig. 5.2(c) shows a very low end-to-end delay in DMAC for very low rate. Even though DMAC is using a staggered sleep schedules, the next-hop node will be awake to receive data. This yields a very low end-to-end delay. When traffic rate becomes, the end-to-end delay increases dramatically. This arises from the increasing buffering delay due to reduced channel utilization. For DGSS, if no packets are detected at the very beginning of each wake-up cycle, the node will sleep. Every packet which

is received during the sleep time has to wait until the next wake-up cycle. As a statistical result, the delay is about a half of the wake-up cycle. Once nodes are awake, all the nodes along the path to sink will keep on until all the packets are sent out. Therefore it won't incur extra end-to-end during wake-up period.

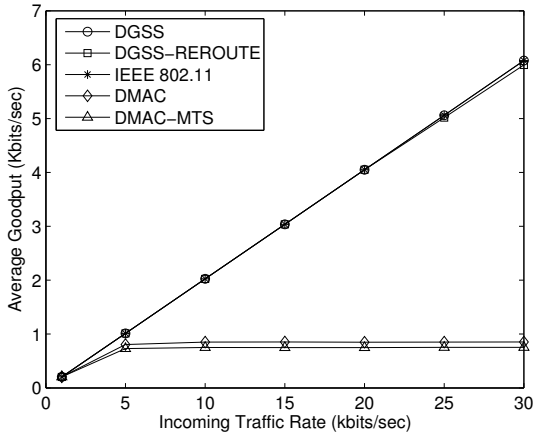
Fig. 5.2(d) shows the average energy consumption at each node. DMAC shows very low energy consumption at low rate, since only the senders and receivers are awake at any give time moment. DGSS and DGSS with rerouting both consume much less energy than the IEEE 802.11, especially at the low rate. However, the energy consumption is still doubled compared to DMAC. DGSS with rerouting consumes little less energy than DGSS at higher rates. It is due to the merge of traffic flows, even though there is overhead for merge and join control packets.

Fig. 5.2(e) shows the number of bits received on every unit of energy consumption. The best energy efficiency is observed for DMAC at lower date rates and it becomes lower for higher data rates. Both DGSS and DGSS with rerouting achieve better energy efficiency than the IEEE 802.11, and their energy efficiency surpass that of DMAC at relatively high data rates. DGSS with rerouting reveals a slightly better energy efficiency at higher rates than DGSS.

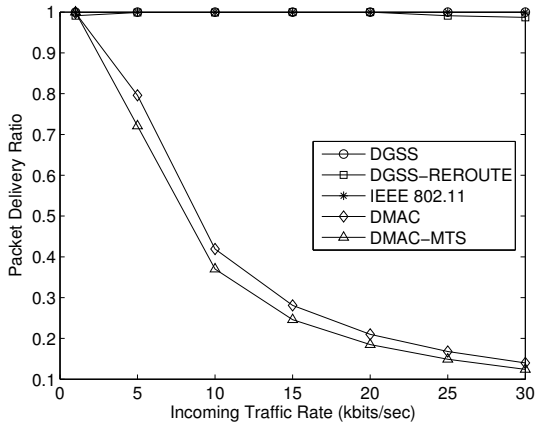
5.5 Summary

In this chapter, we have presented a new sleep scheme for a single-sink data gathering in wireless sensor networks. Instead of staggering schedule, all sensor nodes wake up

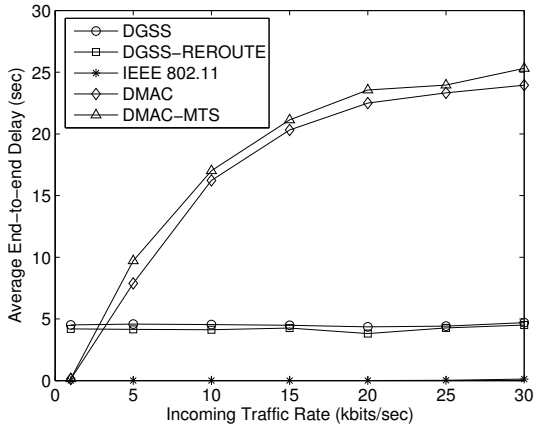
periodically at the same time. The leaf nodes of the data gathering tree rooted at the sink will be turned off first and shrink inward towards the sink node in case of no traffic. Downstream nodes stay awake when traffic load is higher. Simulation results reveal that DGSS gives better throughput and energy efficiency for bursty traffic at relatively high rate than DMAC, while DMAC is still better for the lower-rate traffic.



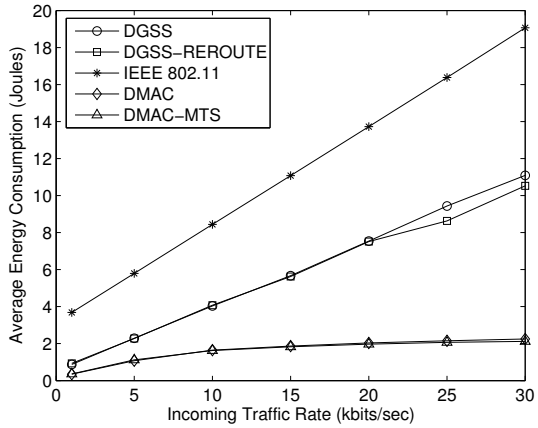
(a) Comparison on goodput



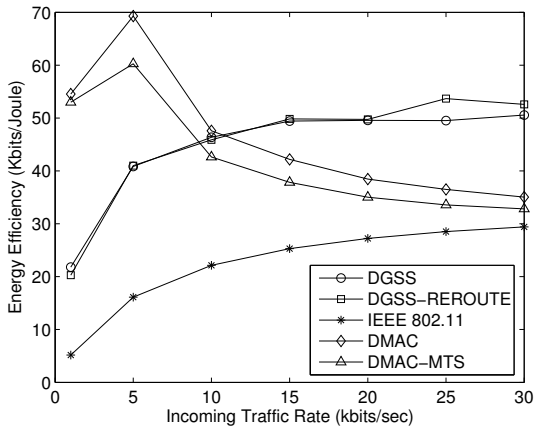
(b) Comparison on packet delivery ratio



(c) Comparison on average end-to-end delay.



(d) Comparison on residual energy



(e) Comparison on energy efficiency.

Figure 5.2: Comparisons among DGSS, DMAC and the IEEE 802.11 with multiple source nodes

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this dissertation, a few research work had been conducted on the design and analysis of energy efficient protocols for WSNs. It is concluded as follows:

- A new MAC protocol, called Pattern-MAC(PMAC), was proposed where patterns in tentative sleep-awake schedules of a sensor node are adaptive to the traffic conditions observed by that node. Patterns are exchanged among neighbors after some time. The actual sleep-awake schedules are generated based on a sensor node's own patterns and its neighbors' patterns. Our simulation results show that in comparison to SMAC, PMAC achieves more power savings under light loads, and higher throughput under heavier traffic loads. Two variants of PMAC — PMAC-I and PMAC-II are proposed to address the tradeoff between energy saving and performance like throughput and latency. PMAC-I gives better power efficiency and energy localization than TMAC and PMAC-II, although its throughput and latency are not as good as TMAC and

PMAC-II. PMAC-II has been proposed in favor of throughput, whose performance is close to that of TMAC. PMAC-I and PMAC-II can be adopted by a single application at the same time and dynamically chosen to fit the needs of the application better. A good balance between energy saving and network performance can be achieved in this case. This also suggests that "pattern exchange" is a promising framework for improving the energy efficiency of the MAC protocols used in WSNs.

- A switch agent was designed at the routing layer for sensor nodes equipped with dual radios. The switch agent sits on top of dual routing agents. It monitors the traffic flow and switches on the high-power radio whenever the traffic load becomes heavy. To save energy while using the high-power radio, the switch agent caches the routes established previously so that a unicast wake-up message can be sent out to selectively wake up the high-power radio at the downstream nodes. The switch agent also keeps a registry for flows which already have the high-power radios awake so that no further wake-up message transmissions are needed for subsequent requests. Simulation results demonstrate that the switch agent yields throughput, delay and packet delivery ratio comparable to the high-power radio interface alone, without incurring much energy wastage. It is of practical values for handling bursty traffic in wireless sensor networks.

- Generalized models of the DMAC protocol were developed for both CBR traffic and stochastic traffic following a Poisson process. A discrete-time Markov chain is used to model the stochastic traffic scenario. The average delay and energy consumption at a source node are expressed and can be evaluated numerically. The close match between the numerical results and the simulation results validates the correctness of the models. The models provide insight on the adaptivity of DMAC under CBR and stochastic traffic conditions and indicate that DMAC only adapts varying traffic conditions to a limited extent due to the low channel utilization and DMAC adapts CBR traffic better than the stochastic traffic.

- A new sleep scheme was proposed for a single-sink data gathering in wireless sensor networks. Instead of using staggered sleep schedules, all sensor nodes wake up periodically at the same time. The leaf nodes of the data gathering tree rooted at the sink will be turned off first and shrink inward towards the sink node in case of no traffic. Downstream nodes stay awake when traffic load is heavy. Simulation results reveal that DGSS gives better throughput and energy efficiency for bursty traffic at relatively high rate than DMAC, while DMAC is still better for the lower-rate traffic.

6.2 Future Work

Most of the work presented here are still tied to a single layer. The energy saving schemes tweaked for one layer may no longer be good once integrated with other layers. Finding integrated and cross-layer optimized energy-efficient solutions remains a big challenge. My future research on wireless sensor networks will be along this line.

Bibliography

[1] Datasheet for chipcon CC2420 2.4GHz IEEE 802.15.4/ZigBee RF Transceiver. http://www.chipcon.com/files/CC2420_Data_Sheet_1_2.pdf.

[2] Wireless LAN medium access control (MAC) and physical layer (PHY) specification, 1997.

[3] The network simulator (ns-2). http://www.isi.edu/nsnam/ns/, 2001.

[4] Standard for part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low rate wireless personal area networks (LR-WPANs), 2003.

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

[6] G. Bianchi. Performance analysis of the IEEE 802.11 distributed coordination function. *IEEE Journal on Selected Areas in Communications*, 18(3):535–546, 2000.

[7] T. Bokareva, W. Hu, S. Kanhere, B. Ristic, N. Gordon, T. Bessell, M. Rutten, and S. Jha. Wireless sensor networks for battlefield surveillance. In *Proceedings of the Land Warfare Conference 2006 (LWC'06)*, October 2006.

[8] M. Buettner, G. V. Yee, E. Anderson, and R. Han. X-MAC: A short preamble MAC protocol for duty-cycled wireless sensor networks. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys'06)*, pages 307–320, Boulder, Colorado, USA, 2006.

[9] N. Burri, P. von Rickenbach, and R. Wattenhofer. Dozer: Ultra-low power data gathering in sensor networks. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN'07)*, pages 450–459, Cambridge, Massachusetts, USA, 2007.

[10] T. Canli, F. Nait-Abdesselam, and A. Khokhar. A cross-layer optimization approach for efficient data gathering in wireless sensor networks. In *IEEE International Networking and Communications Conference (INCC'08)*, pages 101–106, May 2008.

[11] R. Cardell-Oliver, M. Kranz, K. Smettem, and K. Mayer. A reactive soil moisture sensor network: Design and field evaluation. *IJDSN*, 1(2):149–162, 2005.

[12] J.-H. Chang and L. Tassiulas. Maximum lifetime routing in wireless sensor networks. *IEEE/ACM Trans. Netw.*, 12(4):609–619, 2004.

[13] P. Chatzimisios, A. C. Boucouvalas, and V. Vitsas. IEEE 802.11 packet delay - a finite retry limit analysis. In *Proceedings of the IEEE Global Telecommunications Conference (Globecom'03)*, pages 950–954, 2003.

[14] T.-S. Chen, H.-W. Tsai, and C.-P. Chu. Gathering-load-balanced tree protocol for wireless sensor networks. In *Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, pages 8–13, Washington, DC, USA, 2006.

[15] X. Cheng, B. Narahari, R. Simha, M. X. Cheng, and D. Liu. Strong minimum energy topology in wireless sensor networks: NP-completeness and heuristics. *IEEE Transactions on Mobile Computing*, 2(3):248–256, 2003.

[16] A. E. F. Clementi, P. Penna, and R. Silvestri. Hardness results for the power range assignmet problem in packet radio networks. In *Proceedings of the Third International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (RANDOM-APPROX'99)*, pages 197–208, London, UK, 1999.

[17] J. Elson and K. Römer. Wireless sensor networks: A new regime for time synchronization. *SIGCOMM Comput. Commun. Rev.*, 33(1):149–154, 2003.

[18] B. Gao, C. He, and L. Jiang. Modeling and analysis of IEEE 802.15.4 CSMA/CA with sleep mode enabled. In *Proceedings of the 11th IEEE Singapore International Conference on Communication Systems (ICCS'08)*, pages 6–11, Nov. 2008.

[19] Z. Hadzi-Velkov and B. Spasenovski. Saturation throughput - delay analysis of IEEE 802.11 DCF in fading channel. In *Proceedings of the IEEE International Conference on Communications (ICC'03)*, 2003.

[20] J. He, Z. Tang, H. H. Chen, and S. Wang. An accurate Markov model for slotted CSMA/CA algorithm in IEEE 802.15.4 networks. *Communications Letters, IEEE*, 12(6):420 –422, June 2008.

[21] T. Hubler. Worry-free wireless networks.

[22] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Mobile Computing and Networking*, pages 56–67, 2000.

[23] J. Jobin, Z. Ye, H. Rawat, and S. V. Krishnamurthy. A lightweight framework for source-to-sink data transfer in wireless sensor networks. In *BROADNETS'05*, pages 756–766, 2005.

[24] H. Kang, X. Li, and P. J. Moran. Autonomic sensor networks: A new paradigm for collaborative information processing. In *2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing (DASC'06)*, pages 258–268, 2006.

[25] H. Karl and A. Willig. A short survey of wireless sensor networks. Technical report, Telecommunication Networks Group, Technische Universitt, Berlin, 2003.

[26] V. Kawadia and P. Kumar. Power control and clustering in ad hoc networks. In *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications (INFOCOM'03)*, volume 1, pages 459–469, April 2003.

[27] D. Kim and M. Liu. Optimal stochastic routing in low duty-cycled wireless sensor networks. In *Proceedings of the 4th Annual International Conference on Wireless Internet (WICON'08)*, pages 1–9, Maui, Hawaii, 2008.

[28] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon. Health monitoring of civil infrastructures using wireless sensor networks. In *Proceedings of the 6th International Symposium on Information Processing in Sensor Networks (IPSN'07)*, pages 254–263, April 2007.

[29] K. Langendoen and G. Halkes. *Energy-Efficient Medium Access Control*, pages 34.1–34.29. CRC press, 2005.

[30] N. M. Larry King. A weighted-density connected dominating set data gathering algorithm for wireless sensor networks. *Computer and Information Science*, 2(4):3–13, 2009.

[31] J. A. López Riquelme, F. Soto, J. Suardíaz, P. Sánchez, A. Iborra, and J. A. Vera. Wireless sensor networks for precision horticulture in southern spain. *Comput. Electron. Agric.*, 68(1):25–35, 2009.

[32] G. Lu, B. Krishnamachari, and C. S. Raghavendra. An adaptive energy-efficient and low-latency MAC for data gathering in sensor networks. In *18th International Parallel and Distributed Processing Symposium (IPDS'04)*, page 224a, 2004.

[33] X. C. M. Ding and G. Xue. Aggregation tree construction in sensor networks. In *IEEE 58th Vehicular Technology Conference*, pages 2168–2172, 2003.

[34] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, pages 88–97, Atlanta, Georgia, USA, 2002.

[35] M. Maroti, B. Kusy, G. Simon, and kos Ldeczi. Robust multi-hop time synchronization in sensor networks. In *Proceedings of the 2004 International Conference on Wireless Networks (ICWN'04)*, pages 454–460, Las Vegas, Nevada, USA, 2004.

[36] N. Meghanathan. An algorithm to determine energy-aware connected dominating set and data gathering tree for wireless sensor networks. In *Proceedings of the 2009 International Conference on Wireless Networks (ICWN'09)*, pages 608–614, Las Vegas, USA, 2009.

[37] N. Meghanathan. An algorithm to determine energy-aware maximal leaf nodes data gathering tree for wireless sensor networks. *Journal of Theoretical and Applied Information Technology*, 15(2):6–107, 2010.

[38] J. Misic, S. Shafi, and V. B. Misic. Performance of a beacon enabled IEEE 802.15.4 cluster with downlink and uplink traffic. *IEEE Trans. Parallel Distrib. Syst.*, 17:361–376, April 2006.

[39] K.-J. Myoung, S. Y. Shin, H. S. Park, and W. H. Kwon. IEEE 802.11b performance analysis in the presence of IEEE 802.15.4 interference. *IEICE Transactions*, 90-B(1):176–179, 2007.

[40] J. Paek and R. Govindan. RCRT: Rate-controlled reliable transport for wireless sensor networks. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems (SenSys'07)*, pages 305–319, Sydney, Australia, 2007.

[41] J. Park and S. Sahni. An online heuristic for maximum lifetime routing in wireless sensor networks. *IEEE Trans. Comput.*, 55(8):1048–1056, 2006.

[42] P. Park, P. Soldati, C. Fischione, and K. H. Johansson. A generalized Markov chain model for effective analysis of slotted IEEE 802.15.4. In *Mobile Adhoc and Sensor Systems, IEEE 6th International Conference*, pages 130–139, Macau, 2009.

[43] T. Pering, Y. Agarwal, R. K. Gupta, and R. Want. Coolspots: Reducing the power consumption of wireless mobile devices with multiple radio interfaces. In *MobiSys'06*, pages 220–232, Uppsala, Sweden, 2006.

[44] C. Perkins and E. Royer. Ad-hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99)*, pages 90–100, 1999.

[45] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys'04)*, pages 95–107, Baltimore, MD, USA, 2004.

[46] S. Pollin, M. Ergen, S. C. Ergen, B. Bougard, L. V. D. Perre, F. Catthoor, I. Moerman, and P. Varaiya. Performance analysis of slotted carrier sense IEEE 802.15.4 medium access layer. In *Global Telecommunications Conference (GlobeCom'06)*, 2006.

[47] R. Ramanathan and R. Rosales-Hain. Topology control of multihop wireless networks using transmit power adjustment. In *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'02)*, pages 404–413, August 2002.

[48] V. Rodoplu and T. Meng. Minimum energy mobile wireless networks. *Selected Areas in Communications, IEEE Journal on*, 17(8):1333–1344, Aug 1999.

[49] R. Shah and J. Rabaey. Energy aware routing for low energy ad hoc sensor networks. In *Wireless Communications and Networking Conference (WCNC'02)*, volume 1, pages 350–355, Mar 2002.

[50] S. Y. Shin, S. Choi, H. S. Park, and W. H. Kwon. Packet error rate analysis of IEEE 802.15.4 under IEEE 802.11b interference. In *Proceeding of the 3rd International Conference on Wired/Wireless Internet Communications (WWIC'05)*, pages 279–288, Xanthi, Greece, 2005.

[51] V. Shnayder, B. R. Chen, K. Lorincz, T. R. F. Fulford Jones, and M. Welsh. Sensor networks for medical care. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems (SenSys'05)*, page 314, San Diego, California, USA, 2005.

[52] T. Stathopoulos, M. Lukac, D. McIntire, J. S. Heidemann, D. Estrin, and W. J. Kaiser. End-to-end routing for dual-radio sensor networks. In *Proceedings of the 26th International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'07)*, pages 2252–2260, Anchorage, Alaska, USA, 2007.

[53] Y. Sun, O. Gurewitz, and D. B. Johnson. RI-MAC: A receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks. In *Proceedings of the 6th ACM conference on Embedded network sensor systems (SenSys'08)*, pages 1–14, Raleigh, NC, USA, 2008.

[54] H. O. Tan and I. Korpeoglu. Power efficient data gathering and aggregation in wireless sensor networks. *SIGMOD Record*, 32(4):66–71, 2003.

[55] A. S. Tanenbaum. *Computer Networks*. Prentice, Upper Saddle River, NJ, 3rd edition, 1996.

[56] T. van Dam and K. Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the 1st International Conference on*

Embedded Networked Sensor Systems (SenSys'03), pages 171–180, Los Angeles, California, USA, 2003.

[57] J. van Greunen and J. Rabaey. Lightweight time synchronization for sensor networks. In *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications (WSNA'03)*, pages 11–19, San Diego, CA, USA, 2003.

[58] C.-Y. Wan, A. T. Campbell, and L. Krishnamurthy. PSFQ: A reliable transport protocol for wireless sensor networks. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, pages 1–11, Atlanta, Georgia, USA, 2002.

[59] C.-Y. Wan, S. B. Eisenman, and A. T. Campbell. CODA: Congestion detection and avoidance in sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys'03)*, pages 266–279, New York, NY, USA, 2003.

[60] C.-Y. Wan, S. B. Eisenman, A. T. Campbell, and J. Crowcroft. Siphon: Overload traffic management using multi-radio virtual sinks in sensor networks. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems (SenSys'05)*, pages 116–129, San Diego, California, USA, 2005.

[61] H. Wang, J. Elson, L. Girod, D. Estrin, K. Yao, and L. Vanderberge. Target classification and localization in habitat monitoring. In *IEEE Proceedings of the International Conference on Speech and Signal Processing*, pages II–597–II–600, 2003.

[62] R. Wattenhofer, L. Li, P. Bahl, and Y.-M. Wang. Distributed topology control for power efficient operation in multihop wireless ad hoc networks. In *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'01)*, volume 3, pages 1388–1397, 2001.

[63] A. Woo and D. E. Culler. A transmission control scheme for media access in sensor networks. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MobiCom'01)*, pages 221–235, Rome, Italy, 2001.

[64] G. Xing, C. Lu, Y. Zhang, Q. Huang, and R. Pless. Minimum power configuration in wireless sensor networks. In *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'05)*, pages 390–401, Urbana-Champaign, IL, USA, 2005.

[65] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A wireless sensor network for structural monitoring. In *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys'04)*, pages 13–24, Baltimore, MD, USA, 2004.

[66] Y. Xue, Y. Cui, and K. Nahrstedt. A utility-based distributed maximum lifetime routing algorithm for wireless networks. In *Proceedings of the Second International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks*, page 18, 2005.

[67] O. Yang and W. Heinzelman. A better choice for sensor sleeping. In *Proceedings of the 6th European Conference on Wireless Sensor Networks (EWSN'09)*, pages 134–149, Cork, Ireland, 2009.

[68] O. Yang and W. Heinzelman. Modeling and throughput analysis for SMAC with a finite queue capacity. In *Proceedings of the 5th International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP09)*, pages 409–414, Melbourne, Australia, December 2009.

[69] O. Yang and W. Heinzelman. Modeling and throughput analysis for XMAC with a finite queue capacity. In *Proceedings of the IEEE Global Telecommunications Conference (GlobeCom'10)*, pages 1–5, Miami, Florida, USA, December 2010.

[70] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, and S. Singh. Exploiting heterogeneity in sensor networks. In *Proceedings of the 24th International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'05)*, pages 878–890, Miami, Florida, USA, March 2005.

[71] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'02)*, pages 1567–1576, New York, NY, USA, 2002.

[72] W. Ye, J. Heidemann, and D. Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Transactions On Networking*, 12(3):493–505, 2004.

[73] T. Zheng, S. Radhakrishnan, and V. Sarangan. PMAC: An adaptive energy-efficient MAC protocol for wireless sensor networks. In *IEEE International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN'05)*, page 237a, Denver, Colorado, USA, 2005.

[74] T. Zheng, S. Radhakrishnan, and V. Sarangan. A switch agent for wireless sensor nodes with dual interfaces: Implementation and evaluation. In *Sixth International Conference on Broadband Communications, Networks, and Systems (BROADNETS'09)*, pages 1–8, Madrid, Spain, 2009.

[75] T. Zheng, S. Radhakrishnan, and V. Sarangan. Modeling and performance analysis of DMAC for wireless sensor networks. In *The 14th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM'11)*, pages 119–128, Miami, Florida, USA, 2011.

Appendix A

Sample MATLAB Code for Evaluating the Integrals

```
function ans = Ijm(j,m)

    syms f t;

    f=1;

    for i=m-j-1:-1:1

        f=int(f,t,t,5*(j+i-1)+1);

    end



    if(j>0)

        f=int(f,t,0,5*(j-1)+1);

    else

        f=int(f,t,0,1);

    end

    ans=factorial(m-j)*double(f);

    ans=ans/((5*(m-1)+1)^(m-j));
```